LOSSY LIGHT FIELD COMPRESSION USING MODERN DEEP LEARNING AND DOMAIN RANDOMIZATION TECHNIQUES

SVETOZAR ZARKO VALTCHEV

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN MATHEMATICS AND STATISTICS
YORK UNIVERSITY
TORONTO, ONTARIO

JUNE 2022

Abstract

Lossy data compression is a particular type of informational encoding utilizing approximations in order to efficiently tradeoff accuracy in favour of smaller file sizes. The transmission and storage of images is a typical example of this in the modern digital world. However the reconstructed images often suffer from degradation and display observable visual artifacts. Convolutional Neural Networks have garnered much attention in all corners of Computer Vision, including the tasks of image compression and artifact reduction. We study how lossy compression can be extended to higher dimensional images with varying viewpoints, known as light fields. Domain Randomization is explored in detail, and used to generate the largest light field dataset we are aware of, to be used as training data. We formulate the task of compression under the frameworks of neural networks and calculate a quantization tensor for the 4-D Discrete Cosine Transform coefficients of the light fields. In order to accurately train the network, a high degree approximation to the rounding operation is introduced. In addition, we present a multi-resolution convolutional-based light field enhancer, producing average gains of 0.854 db in Peak Signal-to-Noise Ratio, and 0.0338 in Structural Similarity Index Measure over the base model, across a wide range of bitrates.

Acknowledgements

First and foremost, I would like to thank my supervisor Dr. Jianhong Wu, for the time, efforts and resources invested in me, the patience practiced towards my work, and the comprehensive feedback to help polish my research. In addition, I would also like to extend the gratitude to the rest of my supervisory committee, Dr. Hongmei Zhu and Dr. Michael Chen, for your teachings, guidance and aid throughout this journey. Lastly, I would like to thank my mother and father, who I cannot describe as anything less than my biggest fans, and who have supported me every step of the way.

Table of Contents

List of Tables

List of Figures

Abbreviations

AAR Accuracy Above Random Guess

ADR Automatic Domain Randomization

AI Artificial Intelligence

bpp bits per pixel

CA Camera Array

CAE Compressive Autoencoder

CAD Computer Aided Design

CNE Convolutional Neural Enhancer

CNN Convolutional Neural Network

CPU Central Processing Unit

CV Computer Vision

DCT Discrete Cosine Transform

DR Domain Randomization

EIA Elemental Image Array

EPP Epipolar Plane

FoV Field of View

FPC Focused Plenoptic Camera

GAN Generative Adversarial Network

GPU Graphics Processing Unit

GRG Generalized Reduced Gradient

HEVC High Efficiency Video Coding

IDCT Inverse Discrete Cosine Transform

LPIPS Learned Perceptual Image Patch Similarity

ML Machine Learning

MLA Microlens Array

MRP Minimum Rate Predictor

MS-SSIM Multiscale Structual Similarity Index Measure

MSE Mean Square Error

NeRF Neural Radiance Field

OS Occlusion Sensitivity

PSNR Peak Signal-to-Noise Ratio

RCT Reversible Colour Transforms

ReLU Rectified Linear Unit

RGB Red-Green-Blue

SAI Sub Aperture Image

SPC Standard Plenoptic Camera

SRCNN Super-Resolution Convolutional Neural Network

SSIM Structual Similarity Index Measure

STN Spatial Transformer Network

YCbCr Luma-Blue-Red

Chapter 1

Introduction

1.1 Motivation

In the oncoming years, digital content is expected to drastically change. Technological breakthroughs are happening on a daily basis, with the world's biggest corporations like Meta (formerly known as Facebook) gearing up towards release of entire virtual worlds [1, 2]. With the current state of the world, the increase in remote work has only sped up the desire for newer and more immersive means of communication and media consumption. For true immersion, we will need to break free of the static viewport imposed by current media forms, and allow users a full 6 degrees of freedom in terms of head position and stereoscopic view direction, as typically offered in headsets. The light field is one such representation, capable of encoding the light information of an entire scene. This new technology however, comes with it's own set of challenges in terms of content capture, storage, transfer and computational complexity. A central issue (not limited to just light fields) is the large amount of data needed to represent

these life-like media formats. With Moore's Law approaching its limits, we can no longer simply rely on more powerful hardware to overcome the computational problems. New compression methods and standards will surely need to be developed to tackle these issues.

Furthermore, as the technology is still in its infancy there is not much light field data publicly available for research purposes. With the sheer lack of such datasets, much of the current attempts fail to generalize due to inadequate performance metrics on inconclusive sample sizes, often of highly correlated data [3]. Larger datasets will be needed for such compression codecs, both for the aid in the development stage as well as the validation of these proposals. A large, diverse, unified dataset will allow for the streamlining of comparisons of models in the research community, similar to the standards already in place for images and videos in the Computer Vision (CV) community.

1.2 Overview

There has been a great deal of progress in the field of CV in the last decade with the emergence of deep learning and Convolutional Neural Networks (CNNs). Combining local spatial information in patches of images, as well as temporal information in videos, with the high capacity of deep neural networks to approximate non-linear relations was the basis of much of the literature. Utilizing these same ideas, while extending them across angular dimensions appears to be a promising direction for the size reduction problems mentioned. In modern Artificial Intelligence (AI) and deep learning, models are trained to encode feature detectors for specific patterns

and anomalies. These detectors can strategically be used as filters to favour certain patterns over others, as well as to enhance or rebuild particular regions of a given signal.

As most Machine Learning (ML) methods require large datasets (especially neural networks) to learn on, a new dataset is needed. Current capture instruments are designed for the enterprise level, making data collection an expensive endeavour (more on this in Section 2.2). Instead, the idea of Domain Randomization (DR) can be applied to attain such data. Using DR, a diverse dataset can be generated quickly and cost effectively, across a variety of domains, useful for generalization purposes. Such methods have shown promising results, when mixed with real data, or in the absence of it, for the training of CNN models. This will be discussed more thoroughly in Chapter 3.

In this dissertation, we provide evidence for the effective use of data synthesized using DR for the task of image classification in the real world domain. These findings inspire the generation of the LIAM-LF-Dataset; a synthetic light field dataset made of 20,000 samples, including depth and segmentation maps. We introduce a continuous and differentiable function to approximate the integer rounding operation, useful for the training of neural networks. Using the LIAM-LF-Dataset and the rounding approximation, we are able to compute a light field quantization tensor, optimized for reconstruction quality. The quantization tensor along with the addition of a quantization factor is used in the proposed compression method, to present an efficient variable bitrate codec for light fields. We also propose a method to achieve a target bitrate, through the approximation of the associated quantization factor. Lastly, we develop a neural enhancer capable of reducing compression artifacts formed

by the quantization step in the pipeline, and show that our method is capable of reconstructing the compressed light fields with minimal effects on perceptual quality, even at compression ratio exceeding 100:1.

1.3 Outline of report

This dissertation is structured as to progressively take the reader through the key concepts and relevant literature associated with each topic, with an emphasis on a neural based method for the lossy compression of light fields. The reader is encouraged to skip ahead directly to Section 4.5 for details on the pipeline and evaluation of the model.

Chapter 2 will briefly outline the history and formulation of light fields. It will also introduce the concept of DR, as well as other relevant topics and metrics from informational theory and image processing. Chapter 3 is an extensive study of DR and it's application to image classification. We cover an extensive literary review, and present numerous studies to gauge the viability of DR. Next, we extend the strategy into the generation of a light field dataset in Chapter 4, and compare it to other publicly available datasets. This chapter also explores standard lossy and lossless compression methods, and any relevant extensions to deep learning and light fields. We then present a detailed outline of our proposed compression technique, discuss challenges in the training process, and provide statistical and perceptual results. We summarized our findings in Chapter 5, and discuss further avenues worth exploring.

Chapter 2

Background

2.1 Plenoptic Function and the Light Field

The most general form of the plenoptic function describes the light profile in space, direction, time and wavelength [4]. Adelson and Bergen model this as the 7-dimensional function

$$P = P(x, y, z, \theta, \phi, \lambda, t) \tag{2.1}$$

where and x, y, z are the spatial positions of interest, θ and ϕ give the polar and azimuthal angles for the direction of light, λ is a specific wavelength and t is the time. Obtaining such a function is not possible in practice, as it could model the entire universe from the beginning of time. It is however, a good theoretical starting point.

By fixing a specific moment in time, and simplifying our wavelength profile down to a monochromatic model, the plenoptic function can be reduced to the more commonly used 5-dimensional version

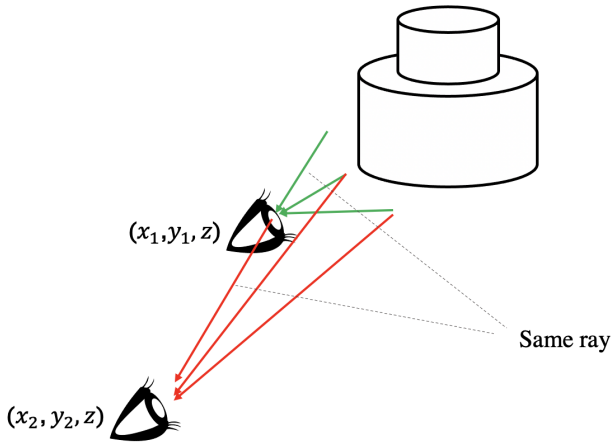


Figure 2.1: In the absence of occlusion, a ray's radiance is constant, and therefore the top rays observed from both vantage points as shown above, are the same.

$$P = P(x, y, z, \theta, \phi) \tag{2.2}$$

Given some closed, reasonably spaced volume, such a function would be able to generate any possible view point, in any direction, hereby analogous to a 3-D photograph. Moreover, since the radiance along a ray remains constant along it's trajectory if it is not blocked, we can further reduce the function to 4-dimensions. This is illustrated in Figures 2.1 and 2.2. The coordinates associated with each ray can be projected to a plane, and parameterized by only 2 spatial variables, along with the 2 angular variables, producing what is commonly referred to as the light field

$$L = L(x, y, \theta, \phi) \tag{2.3}$$

Coined by Gershun in a paper by the very name [5], the concept did not gain much popularity in digital rendering until 1996, when two independent groups presented the

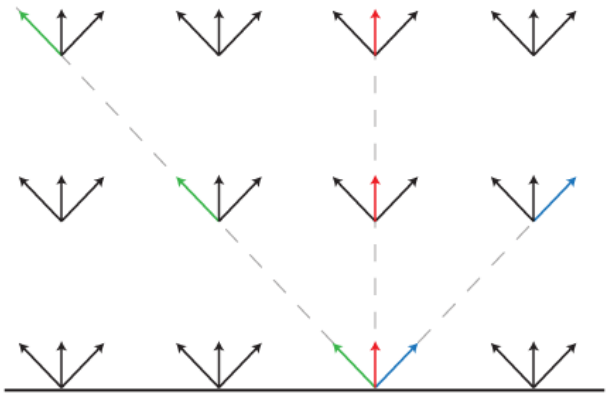


Figure 2.2: Redundancies in the radiance of light can be condensed down along any given ray, such that spatial dimensions can be represented by a projection on a 2-dimensional plane. Illustrated above is the same concept from 2-dimensions projected onto a line (1-dimension).

ideas, interestingly at the same conference. Levoy and Hanrahan and Gortler et al. outlined how a light field can efficiently be captured, parameterized and rendered [6, 7]. Gortler et al. referred to this function as the Lumigraph, while other such as Moon and Spencer termed it the photic field [8] in their works. The term light field has been the one most commonly used today, but the ideas are all fundamentally the same.

While the current formulation of the light field we have defined above can be envisioned as spherical rays along a plane, we are not limited to a flat surface in this matter. Another possible parameterization can be given by two points on the surface of a sphere. This is a less common setup, but may be useful in camera setups for captures, due to the uniform sampling [9]. The more practical and commonly seen form in research is the two plane parameterization referred to as a light slab [6]. We refer to the two as the uv and st planes. All rays can be represented by a line

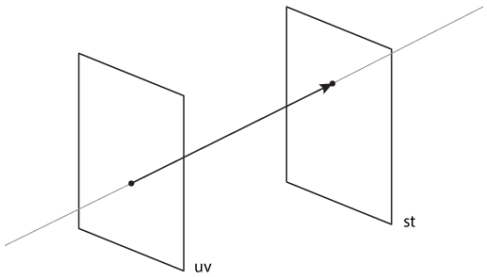


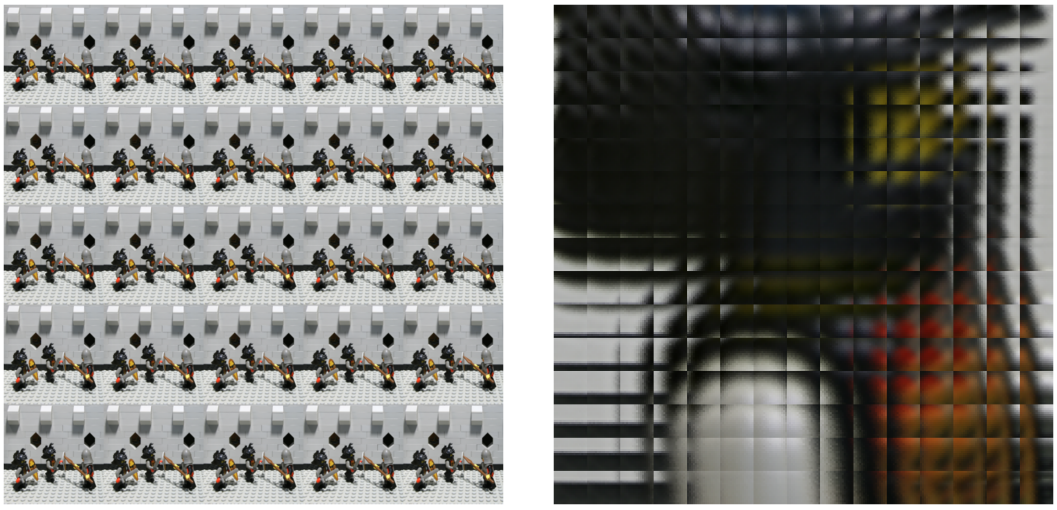
Figure 2.3: A light slab formulation consisting of 2 planes, where rays of light are parameterized by coordinates (u, v, s, t).

connecting points (u, v) on the uv plane to points (s, t) in the st plane. See Figure 2.3. The radiance L is now given by

$$L(u, v, s, t) \tag{2.4}$$

where u, v, s, t are all coordinates on the planes, usually normalized to lie between 0 and 1. By convention, the uv plane represents the angular dimensions (**camera plane**), while the st plane represents the spatial dimensions (**focal plane**) of the light field. This is helpful for light field captures as we will see in Section 2.2, where the two planes will represent the cameras' position in space and their Field of View (FoV) respectively.

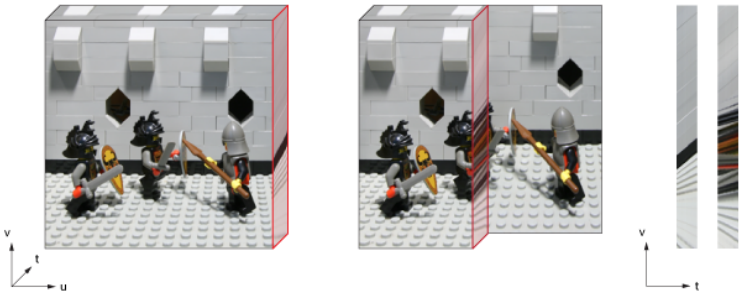
If we were to fix points (u^*, v^*) on the angular plane, the resulting field $L(u^*, v^*, s, t)$ would generate a regular image as we are use to seeing from digital cameras. We will refer to these as **perspective views**, or **Sub Aperture Images (SAIs)**. Alternatively, if we fixed points (s^*, t^*), then $L(u, v, s^*, t^*)$ would represent what any given pixel (s^*, t^*) in an image array would look like from different viewing positions. These



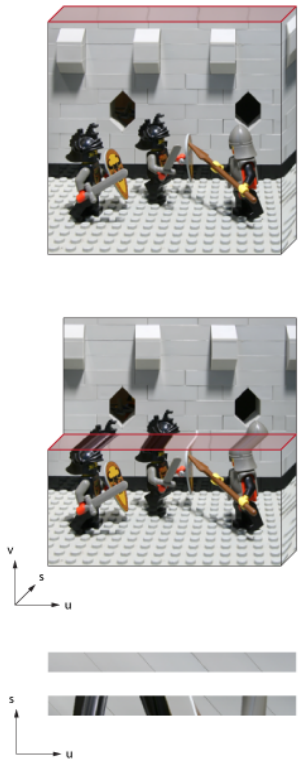
(a) uv plane **(b)** st plane

Figure 2.4: Visualization of light field across both planes. (a) Perspective views across a 5x5 viewpoint sectional. (b) Reflectance views across a 15x15 pixel image block. Light fields are taken from Lego Knights as part of Stanford dataset [10].

resemble reflectance maps and so we shall refer to them as **reflectance views**, or **microlens images** going forward. Sampling the light field $L(u, v, s, t)$ at equidistant spaces, we can think of the above two as slices in tensor. There is one further family of slices, which are important for the geometry of a light field. Fixing (u^*, s^*) and varying (v, t), we can construct an Epipolar Plane (EPP). EPPs display the relationship between angular and spatial dimensions. In this fashion, the slice (in the discrete case) would represent a column of pixels in an image (spatial), and how it changes when the view point is raised vertically (angular). See Figure 2.5a. This creates a parallax effect, which can be used to extract depth information from the scene. Visually, the slope of a given pixel across the EPP is meaningful, as it is directly proportional to depth of that particular pixel in the scene. An EPP can also be sliced across (u, s) keeping (v^*, t^*) fixed, as shown in Figure 2.5b.



(a) Examples of vertical EPPs generated by looking at a fixed column of a light field image, while sliding the view point vertically.



(b) Examples of horizontal EPPs generated by looking at a fixed row of a light field image, while sliding the view point horizontally.

Figure 2.5: A visual representation of EPPs being slices along a light field tensor volume. Light field data taken from the Stanford dataset [10].

2.2 Cameras and Digital Rendering

While the theoretical model for the light field is well formulated, capturing such a function has its challenges and limitations. All light field cameras are based on discrete sampling of the light field, and no devices to date, have been proposed to measure the field in a continuous fashion. Such imaging based solutions take up physical space, and therefore limit the angular and spatial resolutions of the capture. This often leads to a tradeoff between the two, which will depend on the task at hand.

Light field capture was first proposed by Lippmann in 1908 using the idea he termed integral photography [11]. Using an array of lenses which he called an Elemental Image Array (EIA), different viewpoints of a scene could be captured on photographic plates. Adelson and Wang expanded on the idea in the digital age, using a Microlens Array (MLA) inspired by Lippmann's EIA also sometimes referred to as a lenslet. [12]. The camera places a MLA at the typical focal plane behind the main camera lens, with the image sensor just underneath it, as can be seen in Figure 2.6. The sensor will produce an image similar to those seen in Figure 2.4b, typically with some degree of vignetting due to the round shape of the lens. The sensor will impose a restriction on the overall resolution of the entire light field, but the actual spatial resolution will be given by the number of microlenses in the array. In order to capture a light field at an angular resolution of $M \times M$ and a spatial resolution of $N \times N$, and sensor with a resolution of $MN \times MN$ is required.

Though the image in Figure 2.6 displays a grid-like structure of the associated lenses in the MLA, this is not the only way to position them. More commonly seen is a MLA with lenses assembled in a honeycomb pattern, for optimal packing in a cross

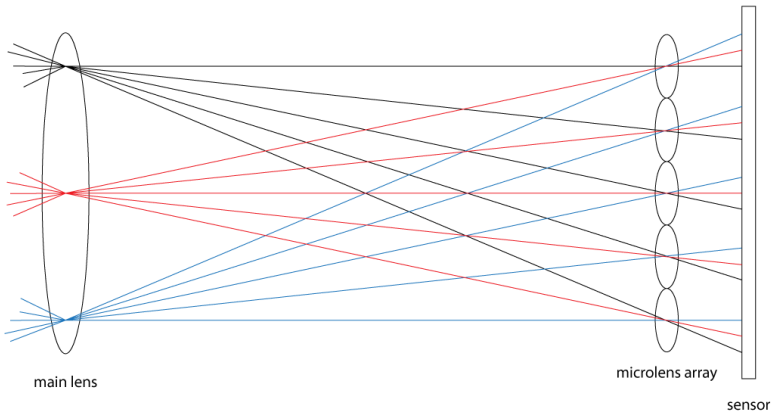


Figure 2.6: Typical architecture of a standard lenslet-based light field camera. Colour is shown to depict where each subaperture ray lands on the camera sensor. By rendering only the pixels off the image captured where a specific colour meets the sensor, a novel perspective view can be generated.

sectional area. These give rise to microlens images in a hexagonal pattern, similar to what can best be described as insect vision. The sampling of the u and v coordinates will be different, but the overall idea is the same.

These types of cameras fall under the class known as Standard Plenoptic Cameras (SPCs). Ng et al. described such a SPC in 2005 [13], which he would later miniaturized and turn into the first consumer ready light field camera called the Lytro. Perwass and Wietzke proposed a design for a SPC with far greater precision in 2012 [14]. They would go on to develop the Raytrix light field cameras, targeted at the industrial and scientific sectors.

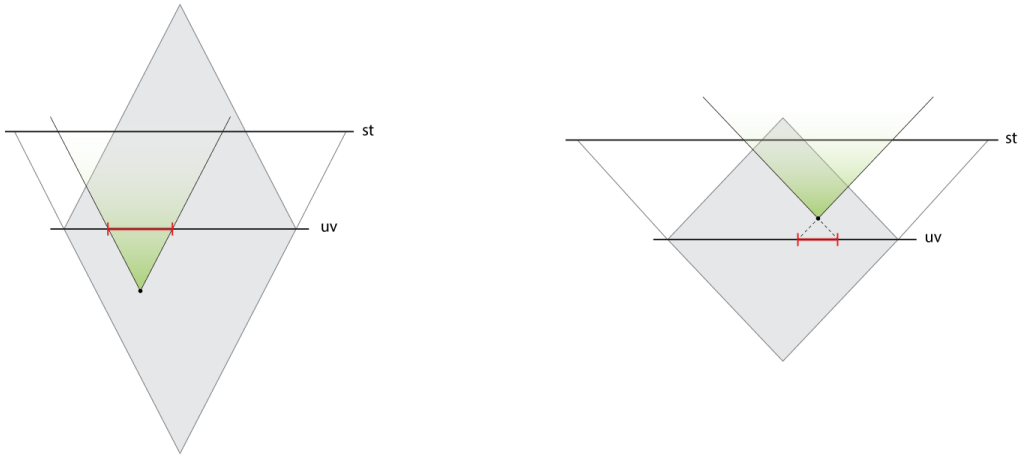
Far more intricate designs have been proposed for lenslet-based cameras, in order to optimize for spatial resolution capture. By sliding the MLA off the main lens focal plane, angular resolution can strategically be sacrificed in favour of spatial resolution. These are usually classified as Focused Plenoptic Cameras (FPCs), as outlined by

Georgiev and Lumsdaine [15]. FPCs are often far more expensive to produce, and are yet to be commercialized.

Aside from lenslet-based cameras, one obvious method of capturing light fields based on the 2 plane representation, is to use a grid of separate cameras, known as a Camera Array (CA). Such an apparatus was discussed and built by Wilburn et al. [16, 17, 18], and was used to capture the images in Figures 2.4 and 2.5. Each camera lies on a point in the uv plane, and its sensor captures the st plane, representing a slice of the 4 dimensional light field. CAs are much more practical and common in research. With high definition cameras being readily available, CAs allow for maximum spatial resolution, at the cost of angular resolution, among all plenoptic cameras. CAs also need not be positioned using the 2 plane parameterization. Aligning cameras strategically in a spherical shape produces light field approximation with a 360 degree FoV. In theory, any view within the sphere can be synthesized using this structure. This is particularly useful for creating immersive virtual reality experiences. Examples of such cameras include the Jaunt One, the Lytro Immerge, the Insta360 Titan, Facebook's Manifold and Google's custom built panoramic system [19].

One final method of capturing a light field, is by translating a single camera in space, as long as the scene is static. Typically, the camera is mounted onto a gantry which sweeps out a plane of perspective views [6, 20], but other methods are also viable [21, 22].

Using the $L(u, v, s, t)$ representation of our light field, we can render views along the uv plane. Specifically, the discrete points which we sampled initially during capture time are already known. Views in between these points, need to be interpolated. However, this is not a trivial task. Nearest neighbour, bilinear and bicubic based



(a) Example of a novel view with a narrow setup in 2D.

(b) Example of a novel view with a wider setup in 2D.

Figure 2.7: Information from the light field can be used to reconstruct any view point in the shaded region. (a) shows a narrow initial capture FoV, with a novel view behind the *uv* plane. Rays for this viewpoint can be taken directly from subsequent views along the view path. (b) shows a wider FoV, with a viewpoint in front of the *uv* plane. In order to render this view, each ray needs to be traced back onto the *uv* plane.

methods all produce ghosting and blurring artifacts due to the averaging out of different views. Geometric methods [23] as well as methods from signal processing [24], optimization [25] and deep learning [26, 27] have been developed for the interpolation of smooth and visually pleasing results, often referred to as angular interpolation or super-resolution. The 4-D light field representation does not however limit view rendering from just the *uv* plane. Novel views can be synthesized off the camera plane, in a region related to the FoV of the initial cameras used in capture. This region can be seen in Figure 2.7. These new viewpoints can be rendered by tracing each ray to the *uv* plane, which in turn can be interpolated up to the desired density using the angular super-resolution methods mentioned.

2.3 Domain Randomization

ML algorithms, particularly neural networks often require large datasets in order to converge to meaningful solutions. In supervised learning, these datasets also require accurate labelling, at times down to the pixel level such as in CNNs. Acquiring such large data samples is a time consuming challenge and can be very expensive. Labelling it can be even more daunting or at times impossible, namely when accuracy is of utmost importance. To combat a lack of data, researchers often have to rely on data synthesis. However, synthetic data can inherently carry large bias and discrepancies to real data. In order to "bridge this gap", Tobin et al. proposed the idea of DR. By randomizing many visual parameters in their simulations, such as textures and lighting, Tobin et al. were able to generate data which they successfully used to train a model capable of robotic control in the real world. Implementing the randomization process led the network to seeing the world as just another variation, concentrating only on the structure and geometry of the scene needed for the task.

Synthetically rendered images have a few advantages over real ones. Firstly, any labelling associated with the render comes for free and with pixel perfect accuracy. For example, object segmentation, depth maps and bounding boxes can all be generated at render time along with the images. Second, they are extremely cost effective to produce, as the scene only needs to be designed once, and each subsequent sample can be renderer for free. This allows for arbitrary large dataset acquisition. Lastly, depending on the level of realism required, they are on average much quicker to produce, than capturing photos in the real world. Together with the labelling time, this task can be more cumbersome for just one real image than for the entire synthetic

set combined. High quality photorealistic renderings can take up to an hour on modern Graphics Processing Units (GPUs), but more typical images can be generated on the order of milliseconds. Graphics software such as Blender and Maya, are capable of creating images indistinguishable from photographs, making use of modern ray tracing and light transport algorithms. On the other hand, game engines such as Unity and the Unreal Engine provide the perfect flexibility for procedural scene generation while being optimized for quick and efficient rendering.

2.4 Compression and Entropy

The process of reducing digital file sizes can be split into two categories. Lossless compression aims to take advantage of any redundancies in the data such that no information is lost. Such a compression will always lead to perfect reconstruction after the encoded signal is decompressed. In order to achieve maximal compression, all redundancy must be removed reducing the encoding to it's most compact form. This limit is usually measured by the Shannon Entropy [29] as

$$H(X) = -\sum_{i=1}^{N} P(x_i) log P(x_i) \tag{2.5}$$

where X is a signal constructed of N symbols x_i, and their respective probabilities are given by $P(x_i)$. The Shannon Entropy can be used as a measure of the compressability of a signal, or conversely the amount of redundancy in it.

Lossy compression on the other hand, goes a step further by removing unnecessary information in order to achieve a balance between quality and compression size.

This process will often vary based on the task at hand. For example, a lossy image compression may choose to toss away high frequency parts of the signal which may lie outside the perceptual range of humans. Such a method is the basis of modern image codecs such as JPEG [30].

The JPEG codec consists of a few different steps. First, the Red-Green-Blue (RGB) image is converted to the Luma-Blue-Red (YCbCr) color space. In the YCbCr color space, the image is factorized into its luma and chromatic components. The human eye is more sensitive to the luma of an image, than it is to the chroma [31]. JPEG takes advantage of this phenomena by subsampling the chroma components, reducing the overall size of the image. The next step in the pipeline is to convert patches of each channel into the frequency domain in 2 dimensions using the Discrete Cosine Transform (DCT) [32] given by

$$c_{i,j} = \sum_{s=1}^{N} \sum_{t=1}^{N} x_{s,t} \cos\left[\frac{\pi(i-1)}{N}\left(s - \frac{1}{2}\right)\right] \cos\left[\frac{\pi(j-1)}{N}\left(t - \frac{1}{2}\right)\right] \tag{2.6}$$

where $c_{i,j}$ is the coefficient of a 2 dimensional cosine signal with periods i and j, $x_{i,j}$ is the pixel in position i, j of the patch, and N is the patch size. In the frequency domain, each patch is divided by some value in a predefined lookup table based on a desired quality level, and the final values are quantized to the nearest integer. This process removes most high frequencies from the patch, leaving mostly 0 entries. The patch can then be strategically encoded, using run-length encoding. While this step is lossless, it is worth mentioning that the quantization discards information, and therefore the entire framework is lossy. The decoder reverses these steps, constructing an image with very low perceptual difference to the initial.

Since lossy compression introduces a degree of freedom in terms of quality, there is no meaningful limit and compression metrics are usually reported empirically at different qualities. The JPEG standard typically produces compression rates of 5:1 for its highest quality, while up to 120:1 for lower levels [33].

2.5 Evaluation Metrics

Media compression and image reconstruction task often require quantitative measures in order to evaluate the proposed methods and compare them to other contenders. Lossy compression in particular, being a tradeoff between size reduction and maintained quality, requires metrics for both.

2.5.1 Similarity Metrics

The most commonly used measures for image similarity are the Mean Square Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) [34]. Given an $n \times m$ target image I with c colour channels, with b bits per colour channel, and an approximation image R we have

$$MSE = \frac{1}{nmc} \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{c} [I(i,j,k) - R(i,j,k)]^2 \tag{2.7}$$

and

$$PSNR = 20log_{10} \left(\frac{2^b - 1}{\sqrt{MSE}} \right) \tag{2.8}$$

with low MSE and high PSNR scores indicative of better quality approximations. Note

that since the PSNR is only a function of the MSE, there is a monotonic relationship between the two, with the PSNR scaled more closely to align with human perceptional measures.

Both of these metrics rely on the absolute error between images, which often contrasts qualitative measures in human experiments. Instead the Structual Similarity Index Measure (SSIM) relies on structural information in a local neighbourhood of an image, producing much more accurate results [35]. The SSIM is calculated on the luma of an image made of M windows as

$$SSIM = \frac{1}{M}\sum_{j=1}^{M}\frac{(2\mu_{I^{(j)}}\mu_{R^{(j)}} + c_1)(2\sigma_{I^{(j)},R^{(j)}} + c_2)}{(\mu_{I^{(j)}}^2 + \mu_{R^{(j)}}^2 + c_1)(\sigma_{I^{(j)}}^2 + \sigma_{R^{(j)}}^2 + c_2)} \tag{2.9}$$

where

$$c_1 = (k_1 L)^2$$

$$c_2 = (k_2 L)^2$$

$$k_1 = 0.01$$

$$k_2 = 0.03$$

$$L = 2^b - 1$$

and $\mu_{I^{(j)}}, \mu_{R^{(j)}}, \sigma_{I^{(j)}}^2, \sigma_{R^{(j)}}^2$ and $\sigma_{I^{(j)},R^{(j)}}$ are the mean, variance and covariance of window j in the target and reconstructed images I and R. Larger values of the SSIM are better, with a score of 1 indicating the images are identical. The Multiscale Structual Similarity Index Measure (MS-SSIM) builds on this idea, by applying the process

across different resolutions of the image, in essence pooling structural information across a variety of scales in the image [36]. Typically, the MS-SSIM produces the most robust results in practice.

Another method of measuring perceptional similarity is through the use of a specifically trained network. Namely, by training a deep CNN against labels of similarity scores attained from human experimentation, a visual perception model can be approximated. One popular such measure is the Learned Perceptual Image Patch Similarity (LPIPS) [37]. Values closer to 0 indicate imperceptible differences, while larger values reflect greater disparities.

2.5.2 Compression Metrics

In order to compare different compression algorithms, it is useful to utilize a standardized metric to measure the overall reduction in a file's size. The most intuitive such metric is the relative size reduction of the file after compression, known as the compression ratio. The compression ratio is often also expressed as a percentage change. It measures the amount of information that has been removed from the compression, relative to its initial size. More commonly used in image compression is the bits per pixel (bpp). Unlike the compression ratio, the bpp is an absolute measure as it is standardized per pixel. In this way, compression of images at different scales can be fairly compared. The bpp is calculated as the total size of the image in bits, divided by the total number of pixels.

The compression associated with the above metrics can be raised arbitrarily high for lossy methods at the cost of quality. In lossless compression however, this is not

the case. A lossless compression must reconstructs the initial signal perfectly, and so imposes a constraint on the compression variable. The minimum bpp that can be achieve in a lossless compression has been shown by Shannon [29] to be the entropy as shown in Eq. 2.5. Methods for such compression do exist, but are not general enough to be suitable for all problems.

Chapter 3

Domain Randomization for Neural Network Classification

In order to develop a useful light field dataset, we first test the validity of DR, and how effective it is. In this chapter, we test how well DR bridges the reality gap to different domains, and which parameters of the process are most important, when training a neural network with the task of image classification.

3.1 Introduction

Recent advances in convolutional based neural networks have opened the door for automation in a wide variety of visual tasks, including classification, segmentation, object tracking, viewpoint estimation and view synthesis. Where once the challenge was to develop a strong model for each of these tasks, today the challenge has jumped to the acquisition of large and diverse datasets to train the models. Zero-shot and

few-shot learning has been making progress in tackling the data size requirement, but solutions there are still very case dependant. They also still have problems with black swan events [38].

The cost of gathering a large amount of data can be very expensive, both in terms of money and time. Pixel level segmentation on an image, for example, can take hours to properly label on a complex enough scene. Instead, the task can be offloaded to a computer, not only in synthesizing the visual data, but also in annotating it [39, 40, 41, 42, 43]. In this fashion, large datasets can be produced relatively quick and cheap, and any labelling comes not only free, but at beyond human-level accuracy.

The use of synthetic data however introduces what is known as the reality gap [28], which as the name suggests, is the inability for it to fully generalize to the real world data, for numerous reasons including textures, lighting and domain distributions. Achieving photorealism in the synthetic data, comes at the price of computational resources and render time [44]. Ray tracing engines can produce images indistinguishable to the untrained eye from a real photo, but may take dozens of hours to render a single image. Instead, realtime renderers like those used by popular game engines are usually used due to their ability to produce large datasets quickly.

In an attempt to narrow the reality gap, DR is introduced to simulate a sufficiently large amount of variations such that real world data is viewed as simply another domain variation [28, 45]. This can include randomization of view angles, textures, shapes, shaders, camera effects, scaling and many other parameters.

DR has successfully shown to aid in the training of networks for object detection, image segmentation, spatial positioning and depth estimation. For our purposes, we will aim to study it's effectivity specifically to image classification. We further perform

a collection of univariate tests in order to 1) examine which parameters are most significant to the process, and 2) gauge the resulting model's ability to generalized to new domains.

3.2 Related Work

To address the reality gap, DR techniques have been explored, including most notably the work of Tobin et al. [28], where they synthesized images of basic geometric objects on a table, in an attempt to estimate their spacial coordinates, such that a robotic arm could pick them up. Their accuracy varied depending on domain parameters, achieving errors as low as 1.5 cm on average in terms of object location, showing promise for synthetic data training. Notably, they found that the number of images and the number of unique textures used in the images were the most prominent parameters to model accuracy. Camera positioning and occlusion also had meaningful contributions, while the addition of random noise in images did not.

In another work, Tobin et al. [45] discussed the use of the DR, to the objects themselves, again in the aim of robotic grasping. This time, they procedurally generated millions of object meshes, leading to shapes not typically seen in the physical world, with the goal of the model generalizing specifically to the motion of the grasp. When bridged with real world objects, their results show the robot is able to grasp the subject with an 80% accuracy. Again, the number of objects was a contributing factor to the overall accuracy.

Loquercio et al. [46] used DR to bridge the gap between the artificial world and the real one, in the task of autonomous drone flight. In their work, they synthesized

arbitrary race courses for the drone to learn to fly in, and then tested their controller in arbitrary track configurations in the real world. They achieved near perfect course completion scores for many variations including max speed constraints up to 10m/s, and lap totals less than 3. Other results are also substantially higher than other baselines they compared to. Parameters tuned include scene textures, gate shapes, and lighting conditions, with all 3 providing improved results. Similarly, Shafaei, Little, and Schmidt [40], and Atapour-Abarghouei and Breckon [41] showed that depth estimation in general can be well approximated using neural networks trained entirely on synthetic images. They do note however, that having pixel-perfect annotations lead to problems when generalized to a real world domain.

Tremblay et al. [39] outlined a variety of different parameters associated with the DR process in the problem of object detection, with results indicating more parameters give rise to better accuracy, even if only marginally. Furthermore, they noted that freezing the weights of the feature extractor part of the network resulted in worse results, which contrast results obtained by Hinterstoisser et al. [47]. Peng et al. [48] also highlighted this notion, by lowering their learning rate, as to further allow the feature extractor itself to generalize to higher level features, which may only be present in the synthetic data. Their results indicated that viewpoint variation of the objects is far less important in terms of model accuracy than one might suspect, while the model is most sensitive to the amount of unique instances of objects per class.

Another common computer visual task is the problem of viewpoint estimation. Movshovitz-Attias, Kanade, and Sheikh [44] explored this using carefully constructed synthetic images, leading to results nearly as good as real images.

A natural place to utilize synthetic images is in the problem of human pose

estimation. Rigging of joints is an expensive and cumbersome tasks, and generating large datasets for this are nearly impossible. Chen et al. [49] show that training with synthetic images for the task, outperformed training on real images.

Pepik et al. [50] highlighted how CNN accuracy doesn't necessarily transfer across different domains, providing evidence that the networks are not invariant to domain changes. They concluded that training them with mixed data spread across many domains increases generalized performance, similar to the findings of Peng et al. [51].

Improvements to regular DR have been shown when the sampling methods and parameter distributions are carefully adjusted as show by Mozian et al. [52] and Mehta et al. [53]. Prakash et al. [54] were able to apply such methods successfully in the 2D bounding box problem. The team at OpenAI managed to train a robotic hand to solve a Rubik's Cube by generating progressively more difficult environments in what they called Automatic Domain Randomization (ADR), effectively updating their sampling during training [55].

The common findings between most previous research into synthetic data appears to be that it is best used as a supplementary part of the data gathering process, as oppose to entirely relying on it to train a model from scratch. Generally, a mixture of real world data, spread across a broad range of domains, in unison with computer generated imagery tends to produce the best results [56].

Literature in DR for image classification is lacking. We aim to address this shortcomings, and apply similar analysis to its application in terms of accuracy to real world data, as well as its transferability to other domains. We also explore the effects of different DR parameters in Section 3.4.2.

3.3 Methods/Experimental

To generate our images we make use of the popular game engine Unity. We make this choice as we aim to produce a large amount of visual data at a reasonable tradeoff relative to the rendering time required. A game engine in particular is great versus photo-rendering software since it can produce many frames per second, at the cost of photorealism. Since one of the central inspirations to using synthetic data is the fact we can generate lots of data quickly, it does not make sense to spend hours rendering each image to cinematic quality, as this may be even more costly than gathering the real image data by hand.

DR aims to produce data samples spanning as much of the image space as possible. The samples need not follow any distribution observed in the real-world, thus possibly producing many extreme outliers. Consider, for example, how an autonomous car might react when it comes across a car accident on the road, having been trained only on data of clean law-abiding agents. It is evident how valuable such outliers would be at train time. In the case of our classifier, we aim to train our model to detect precisely what features identify each class, regardless of how feasible each sample is.

For our analysis, we begin by importing 3D generated models of cats and dogs from the Unity Asset Store. We proceed to build a random scene with one of these models rendered each frame. First a plane is created, on top of which a model of one of our classes is placed. Our models come rigged with a variety of different animations, and so we randomly sample a frame from a random animation for each subject. This gives us a variety of different poses, sampling across what we can think of as a posture manifold across the image space. Similarly, any other simulation parameter can be

Figure 3.1: Examples of synthetic data generated using DR of cats and dogs. Materials, breeds, skyboxes, camera, lighting, pose and obstructors are randomized across images.

interpreted as distinct manifolds, allowing for sampling across samples that should be invariant in terms of classification. Such sampling should encourage our network to generalize better to the specific features that define each class, as oppose to other latent features. Next, we place our camera at a random position in the upper hemisphere of the plane, keeping a reasonable maximum distance to the subject. The camera is rotated towards the subject with slight perturbations to simulate random locations on the rendered image. Lighting of the scene is randomized in terms of intensity and direction, and a variety of skyboxes were used to generate a random sky. Random 3D volumes are spawned near the subject, at random scales, locations and orientations. We limit the amount of occlusion these volumes can have in a frame as to avoid producing any samples where the entire subject is hidden. Lastly, all objects in the rendered viewpoint are given a random texture. For our data generations, we used 74 different materials including rocks, woods, metals and even unrealistic textures such as sprinkles. We refer to all of these augmentations as the parameters of the synthesizer. Other augmentations such as brightness and saturation adjustments, noise addition and image deformation are excluded from the setup, as they are often implemented at train time.

The entire scene described above is generated every frame and a 256x256 snapshot is save in JPEG format. Examples can be seen in Figure 3.1 Our current setup runs around 20 frames per second, making the image synthesis very efficient, as it allows us to generate 1000 images in less than a minute.

When we refer to randomization in our simulation, we consider a uniform distribution for our subsampling. For example, any location of the camera is equally likely to be chosen, and so we inherently achieve a flat distribution which may not be present in a real world training set. Photographers tend to capture images in canonical views and so this may produce a very bias sample with many head-on and profile views, and not much in between. By using a uniform distribution, we encourage the network to learn the general structure of each class, as oppose to possibly overtraining it to a specific viewpoint, and failing to generalize to cases where a novel viewpoint is encountered. We apply this across all simulated parameters.

3.4 Results

Under the conditions that synthetic data is cheaper and easier to obtain than authentic real world data, we turn our focus to the quality. More precisely we want to study the effect of synthetic data on the overall accuracy of a convolutional classifier. In order to have a fair comparison, we will keep the number of training samples the same. We will also use the same real world test data for both.

We use the Kaggle Dogs and Cats Dataset [57] for the real images. We gather 10,000 train images and 2,500 test images for dogs and cats each, for a standard 80-20 split. We also proceed to generate 10,000 synthetic images of each category as well

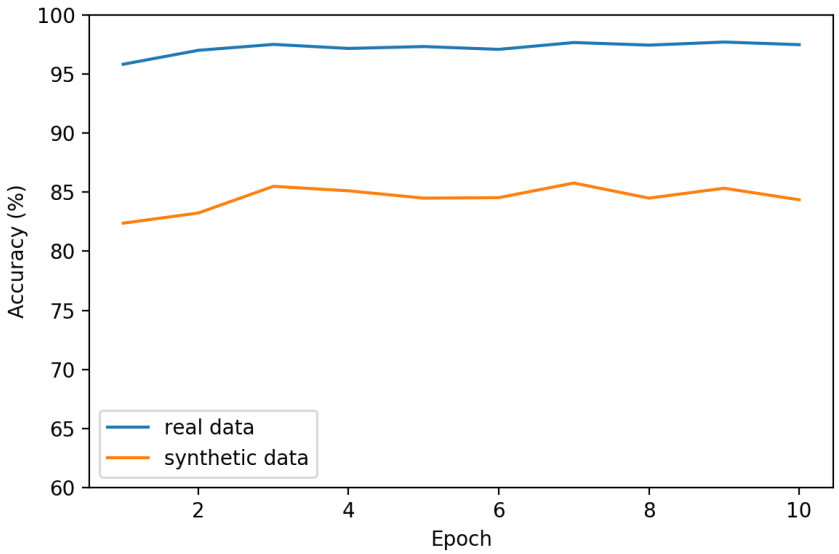


Figure 3.2: Test accuracy of the model using real training data vs synthetic training data over 10 epochs.

using the methods outlined in Section 3.3. In order to keep network architecture choices from effecting our results, we will proceed with a basic pre-trained 16 layer VGG-16 model and stack one extra 128-neuron dense layer on the end of it before making a final classification. This way all feature extraction is done by a standard visual feature extractor and the test is not biased particularly to the task at hand. Standard data augmentation techniques are applied on images at train time, including random cropping, zooming, horizontal flipping, and slight rotation and brightness adjustments. Training the network on a single NVIDIA Tesla K80, with a learning rate of 0.001 over 10 epochs, produces the results in Figure 3.2. Since we have a large set of training data, our model converges quickly. We can notice that the real model achieves a state-of-the-art accuracy of near 97.5% while the synthetically trained model underperforms at 86%.

3.4.1 Network Architecture

Since the inspiration behind DR relies on the idea of model generalization across different domains, it may be of interest to see how the model accuracy will be effected by implementing dropout into the network (0.3 dropout rate), and unfreezing the weights of the VGG-16 block of the network as mentioned in [39] to allow for full model finetuning. Results of optimal accuracy through 10 epochs of training for each can be seen in Figure 3.3. Dropout seems not to have any noticeable effects on the accuracy of the model, while unfreezing the VGG-16 weights seems to have reduced accuracy substantially, in line with results from [47] and [48] suggesting that only the final layer should be finetuned in practice. This could likely be explained by the fact that the VGG network is very large and has taken tons of computational power to train, whereas we only terminate training after 10 epochs. This can be explored further, but will likely require large amounts of resources.

3.4.2 Domain Randomization Parameters

It would be valuable to know which parameters in our DR are most important in the quality of the data, and to what degree. To study this, we train the same model above on a variety of different synthetic training datasets. Each set varies only in the inclusion/exclusion of a DR parameter, while keeping all others present as the control case in Section 3.4.1. This way we can gauge the importance of each parameter in its overall quality contributions to the model accuracy.

We generate 7 new datasets, each omitting obstructors, textures, lighting, and sky boxes, keep subjects in default position as oppose to randomizing their pose, and omit

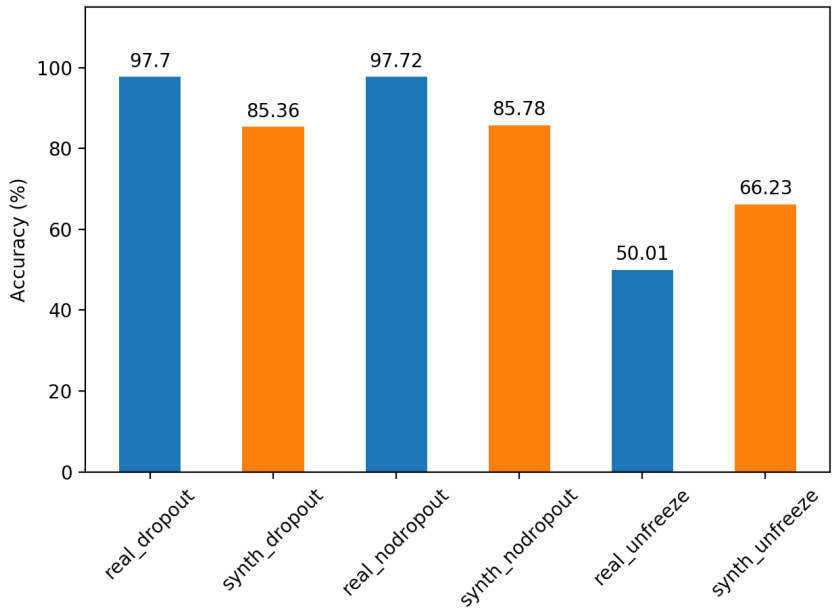


Figure 3.3: Accuracy of the real and synthetically trained models using dropout, no dropout, and allowing for the VGG weights to be trainable (unfreezing).

breed variations, as well as making unrealistic subjects by applying random textures to them (eg. wooden dogs). Results are summarized in Figure 3.4. The skies and the textures to the surrounding seem not to have significant effects on model accuracy. Texturizing the subjects intuitively should generalize better to different domain, but in the case of real world subjects, it seems to actually lower accuracy a bit. The largest significance was caused by the variety of different breeds. Initially we had 10 cat and 28 dog models. Limiting to just 1 of each reduced our accuracy by nearly 16%. What this shows is that the breed variation is the most important factor in learning differentiating features. This could also suggest that increasing the amount of breeds could increase model accuracy to levels near the real model. A surprising result however, was the fact that removing obstructors and keeping the subjects in their default upright poses increased model accuracy nearly 3%. This indicates that

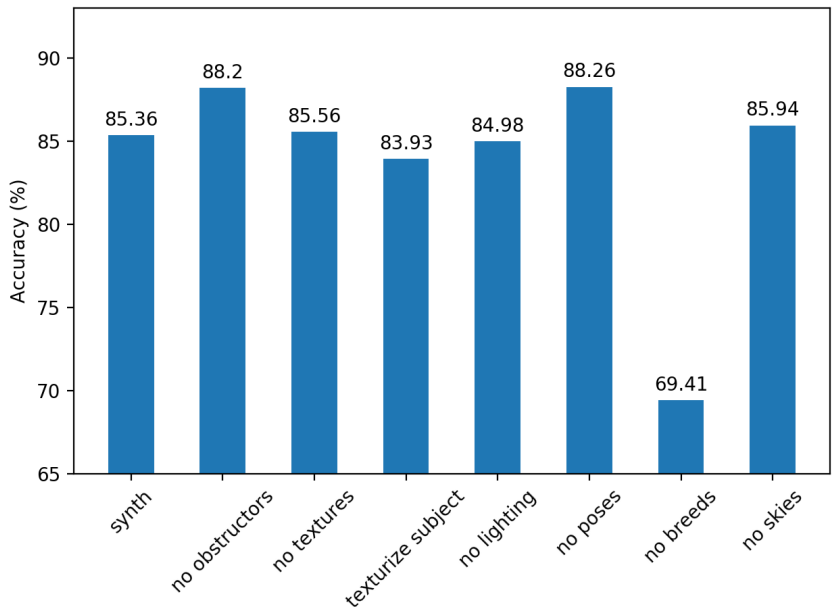


Figure 3.4: Optimal accuracy of models trained using different synthetic datasets. "Synth" represents the control model trained on DR techniques outlined in Section 3.3 using dropout. The others represent datasets were a certain parameter was omitted in the data generation processes such as lighting or textures. We also go a step further and texturize the cats and dogs in what is labelled as the "texture subject" bar.

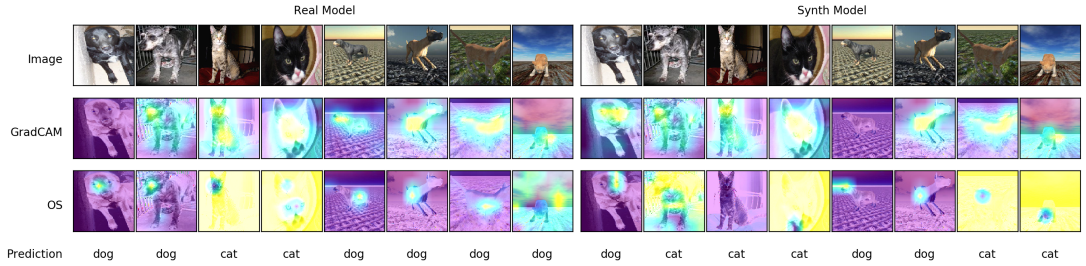


Figure 3.5: First 4 real and synthetic images in each category, and their subsequent analysis as to what each model is looking at to classify each image (real model on the left, synthetic on the right). Second row shows the GradCAM explainer while the third row utilizes Occlusion Sensitivity (OS), when testing for the cat class.

some poses could actually be hiding important features needed to classify each object. The obstructors on the other hand could be blocking parts of the subject or causing unnecessary shadows on the subject, hiding important classification information. This could also indicate that our real test data has a highly condensed distribution in terms of subjects being displayed very clearly in full, and generally being photographed staying still as oppose to being caught mid-action in a certain motion.

3.4.3 Heatmaps

Being mostly a black-box, neural network based methods are still susceptible to overfitting and other unforeseeable errors. For this fact, interpreting our model is another facet worth exploring before we can conclude anything concrete about its accuracy. For this, we apply two state-of-the-art heatmap based techniques to some images from both domains, and inspect which regions of the image our networks value most with respect to their final classification. Some non-handpicked examples can be seen in Figure 3.5. First we apply GradCAM [58], to our images, and specify the cat category as our target class. For the most part, our network tends to focus on

the facial features, including ears and eyes mostly to determine its prediction, in line with what we expect. Notice in Figure 3.5 how each model tends to mistake dominant features in images outside its domain. We explore this further in Section 3.4.4. The last row shows a heatmap of the output produced using the OS method [59]. The results here seem to be more global, with a slight tendency to value a single eye of the subject.

3.4.4 Domain Transfer Accuracy

One property that a domain randomized dataset may provide over a real one is an inherent ability to generalize to other domains. Since the real dataset is obtained from a real world distribution and real world conditions, it is not surprising that it outperforms the synthetic set. However, it is of interest to see how such models will perform across domains it has not seen before. In this section, we test the best models we built thus far on a new small test set that consists of cartoon styled images of dogs and cats (see Figure 3.6). In theory, if our models are truly learning the correct set of features that separate cats from dogs, this task should produce results similar to the initial test set accuracy. Results from this test can be found in Figure 3.7.

Contrary to our hypothesis, the synthetically trained model actually performs significantly worse on this new "Toon" domain vs the real model, by nearly 20%. However, what is interesting is despite the real model's ability to transfer reasonable to the animated domain, it generalized very poorly when tested on the synthetic domain. This suggests that domain generalization is domain specific. A model may transfer well to another domain, while being inadequate on another. In the case of measuring

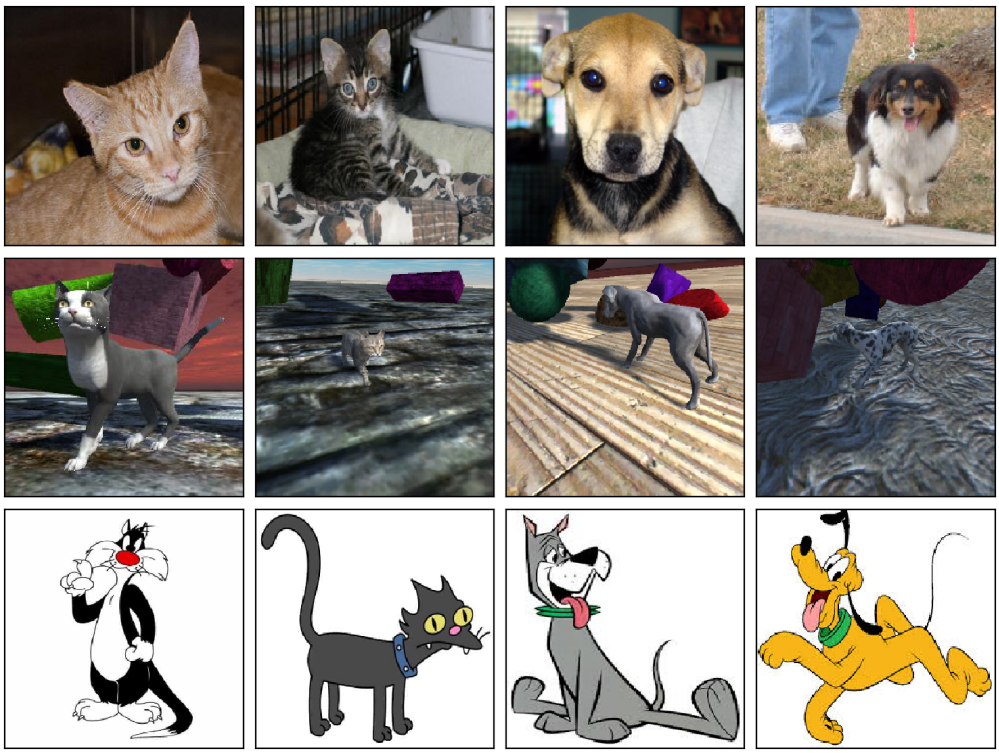


Figure 3.6: Example of cat and dog images shown to our models across different domains. First row is of real world domain images. The second row shows a computer generated 3D domain similar to video games. Third row is a new animated based domain unseen to both models during training. Cartoon images are intellectual property of Warner Bros., Fox Broadcasting Company, Hanna-Barbara Productions and Walt Disney Productions.

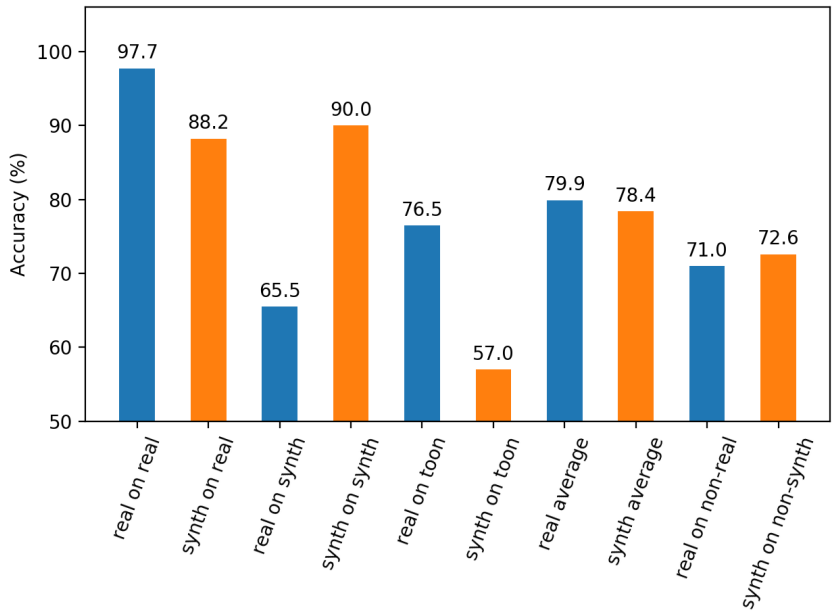


Figure 3.7: Optimal accuracy of the real and synthetic model across a variety of different domains.

out-of-domain accuracy, our synthetic model actually outperformed the real model on average, suggesting that DR does in fact help with generalization.

3.4.5 Increasing Categories

So far, we have only looked at classification accuracy across 2 categories, namely cats and dogs. It would also be of interest to see how our synthetically trained models perform as the number of categories increases. We combine the GRAZ-02 Dataset [60], with the MaviIntelligence Bike Dataset [61] and the StanfordAI Cars Dataset [62] for real world images of cars and bikes, and also incorporate 24 bike and 16 car 3D Computer Aided Design (CAD) models into our data generation process. We limit our studies to 4 categories, namely the cats, dogs, cars and bikes categories, as

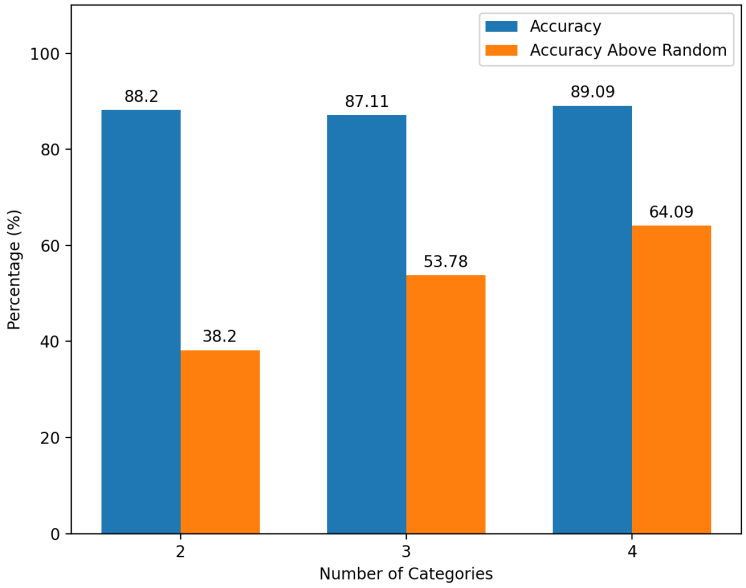


Figure 3.8: Accuracy of the model trained on synthetic data as the number of categories in the classification varies. The 2 category model trained between cat and dogs, the 3 category introduced bikes and the 4 category added the cars class.

gathering 3D models for data generation purposes is a challenge in its own.

Our combined dataset now contains thousands of images of each class, where we again utilize a 80-20 train-test split. Results can be found in Figure 3.8. Overall classification accuracy seemed to remain consistent moving up to 3 and 4 categories. Alternatively, an Accuracy Above Random Guess (AAR) is calculated by

$$AAR = Accuracy - \frac{1}{n} \tag{3.1}$$

where n is the number of categories. Looking at this measure which more accurately accounts for increases in labels, our networks' performance rose with each subsequent category added. One might expect for the overall accuracy to begin to drop as the

categories increase, while the AAR remains constant. However, this is not what we see, suggesting that models trained on synthetic data get better as the number of classes increases.

3.5 Conclusion

Using DR and the Unity Game Engine, we simulate a variety of parameters including breeds, lighting, textures and occluders to generate a vast range of cat and dog images. With this synthetic dataset, we were able to train a classifier to identify photos of real world dogs and cats at an accuracy level of 85.26%. We were further able to increase the accuracy of the model (surprisingly) to 88.26% by actually limiting the posing randomization of the subjects. This was likely due to producing a distribution that was more similar to that of the real world test dataset. It was also found that the most important parameter to randomize is (unsurprisingly) the amount of breeds for each category.

Visual heatmaps suggest that our network is in fact looking at reasonable sections of the image to determine model predictions. Given that our model is trained on synthetic images, it generalized to out-of-training domain better than the model trained only on real world data, scoring an accuracy of 72.6% on a mixture of real and animation images.

Lastly, the synthetically generated data was able to maintain high accuracy when the number of categories for the classifier was increased.

Overall our results are in line with the current literature in other visual based tasks when it comes to the use of synthetic data in training neural networks. The findings

suggest that despite not being able to outperform models trained on real images, the computer generated data can achieve comparable results with the added benefit of being easier and cheaper to attain and scale. Furthermore, as Schraml [56] mentioned, synthetic data should be used in unison with real data, as oppose to a replacement.

Since our initial goal was to compare the effectiveness directly of synthetic images vs real ones, we did not spend much time redesigning the network architecture itself, but focused specifically on finetuning the weights and dropout probabilities. It is possible that the reality gap can be further reduced, given some adjustments to the model itself. Optimal network structure for the generalization of synthetic data to the real world domain, would be an interesting topic worth exploring.

Nevertheless, DR shows promise with regards to data acquisition for a variety of task, and appears to be a viable solution for the light field compression model in the following chapter. We revisit DR specifically in that context, in Section 4.4.

Chapter 4

Convolutional Neural Networks for Lossy Light Field Compression

4.1 Introduction

Current hardware and bandwidth limits make light fields difficult to store, transmit and render due to their inherently large file sizes. Even with perfect lossless compression, the data requirements are still orders of magnitude too big. A lossy light field compression algorithm is needed, that is capable of achieving practical bitrates with minimal perceptual loss in quality, while also being fast in terms of encoding and decoding. In what follows, we outline current academic literature related to the compression of light fields. We then generate a large light field dataset based on the ideas of Chapter 3, made of 20,000 samples, each consisting of 81 SAIs. A shallow convolutional network is used to approximate a quantization tensor used in the compression of the light fields. A second deep convolutional network is trained to further enhanced the compressed

light field reconstruction post-quantization in Section 4.5 using the synthesized dataset. Lastly, we compare the model compression rate and distortion to the state-of-the-art codecs currently available.

4.2 Related Work

4.2.1 Standard Compression Methods

Light field compression literature is still in its relative infancy. Light field and plenoptic media standards have been proposed such as those by the JPEG group, but there is still no generally accepted format. The JPEG Pleno framework introduces 2 independent modes, namely the Prediction Mode, and the Transform Mode [63]. The Prediction Mode is based on geometrical image warping, but requires knowledge of the depth of a scene. Alternatively, the Transform Mode extends the idea of the DCT to 4 dimensions, followed by a combination of optimized decision trees and arithmetic encoding to compress the file. Such a framework produces promising results, but require a high angular view density. Furthermore, compression and decompression speed and hardware requirements are not mentioned.

Standard JPEG compression can be applied independently to each SAI, but fails to take advantage of angular redundancies between views and serves only as a baseline. Methods used in video codecs such as High Efficiency Video Coding (HEVC) have been proposed, that aim to fix this by arranging adjacent light field views in a pseudo-sequence, and treating them as frames in a video [64, 65, 66, 67, 68].

Santos et al. [69] present a detailed analysis of how different colour transformations

effect the overall size reduction when lossless compression is applied to light fields. Their results unanimously suggest that the Reversible Colour Transforms (RCT) allow for the most compact representation, regardless of compression algorithm used, and go on to introduce their own Minimum Rate Predictor (MRP) model. A MRP is based on assigning bits in proportion to an associated estimator's prediction error. In another study, they further improve their MRP model by utilizing information across both microlens views and SAIs [70], minimizing local redundancy. Stepanov et al. [71] propose a learning-based lossless compression, in which they synthesize views and use them as a proxy for the probabilities in an adaptive arithmetic encoding.

When perfect data retention can be sacrificed for significant size reduction, a few reasonable attempts have shown promise. Chen, Hou, and Chau [72] were able to reduce bitrates close to 50% while maintaining reasonably high PSNR values, by carefully encoding disparity information with some optimized key views of the light field. Another method by Tabus, Helin, and Astola [73] used the depth information, along with a set of reference views to reconstruct the other dependant views. Zhang et al. [74] proposed an alternative method encoding the view maps of points on the associated point cloud. They represented each point by a combination of B-Spline wavelets, and further compress each of the wavelet coefficients spatially. Their results achieve PSNR values near 30 dB depending on compression parameters, with computational times varying between 5 and 24 seconds. Similar to the DCT and wavelet formulations, dictionary based methods attempt to create a basis optimized for light field patches. Marwah et al. [75] refers to these as light field atoms; essential building blocks of natural light. In their work, they construct a large dictionary of atoms, capable of sparsely encoding reflectance views. The concept is taken a step

further by Miandji, Hajisharif, and Unger [76], who extend the light field atoms across spatial and angular dimensions. Jiang et al. [77] attempt to calculate the homographies which best align the light field views, achieving a more compact formulation for a desired bitrate.

4.2.2 Learning Based Compression Methods

When it comes to learning based methods, most methods utilize some form of view synthesis, achieving compression by storing only a small subset of SAIs, and reconstructing the lost ones. Zhao et al. [78] and Zhao et al. [79] make use of CNNs to recreate the entire light field using a sparse set of views. Bakir et al. [80] uses HEVC to encode 16 such views, in a similar study, achieving bitrate gains of 30% over standard video codecs. Computation times were not outlined. Gupta et al. [81] go one step further and rely only on a neural network made of 2 separate branches. The first branch uses a fully connected autoencoder, producing optimal angular results, while the other stacked 4D CNN layers leading to improved spatial results. The two branches were combined, achieving PSNR scores between 26-32 dB, whilst attaining a compression ratio of 16:1, but coming at the cost of computation time, logging processing times on the order of minutes.

Singh and Rameshan [82] structure their networks as many different branches of Compressive Autoencoders (CAEs), one for each row of SAIs, presenting results comparable to HEVC while being able to adjust the rate to distortion tradeoff as desired. Unfortunately, most of their calculation take minutes to hours to compress on a standard Central Processing Unit (CPU). Similar CAE networks have been

proposed [83].

Finally, Generative Adversarial Networks (GANs) have also been used to synthesize views from a small set of key views [84, 85]. GANs use a set of networks, trained in unison. The generator network attempts to produce a view as indistinguishable from the real as possible, while the discriminator network attempts to distinguish real from fake. When the two networks can be trained as to successfully converge, results are often very visually pleasing [86]. Wang et al. [87] combines such an architecture with a Spatial Transformer Network (STN) in an attempt to approximate an affine transformation between adjacent light field views, such that only a subset of them are needed during compression.

4.2.3 Other Related Work

The idea of view rendering and novel view synthesis is at the heart of the light field framework. One idea that has been very popular in recent literature and CV conferences is the Neural Radiance Field (NeRF) [88]. A NeRF is a neural network capable of approximating a scene by a continuous volumetric function, including lighting directions. For any given location in space, NeRFs are able to accurately predict the light that would be emitted at said point. In order to achieve high frequency detail in the renderings, the authors introduced the concept of positional encodings [89], by increasing the dimensionality of the spatial and angular dimensions. Much research has gone into improving NeRFs with regards to higher quality, increased convergence times, faster rendering and extensions to more difficult scenes [90, 91, 92, 93, 94, 95, 96]. Unfortunately, NeRFs are implicit representations of a scene, meaning

a new model has to be trained for any given scene. Nevertheless, the representation does encode a light field, and is analogous to a compression method as long as the network size is smaller than the initial light field file.

An alternative form of light field compression can be achieved using image super-resolution. Spatial super-resolution allows for the storing of fewer pixels per SAI, and upsampling during decoding. Fan et al. [97] and Yuan, Cao, and Su [98] demonstrate this using a mixture of CNNs, achieving reasonable qualitative and perceptual metrics on the reconstructions. By arranging SAIs horizontally and vertically, Wang et al. [99] improves these results, using a bidirectional recurrent CNN, treating the images as temporal sequences. Super-resolution can be further extended to the angular dimensions, allowing for more sparse encodings in both overall pixel density, and total number of views [26, 100].

If we omit the angular information in a light field, much of the literature on image compression using deep learning can be applied directly to each SAIs. Alexandre et al. [101] use a convolutional CAE, with a quantizer and importance map blocks, capable of assigning bit counts per pixel as it deems optimal. In order to backpropagate thought the quantization layer, they use a smooth gradient approximation to the floor function which they call soft-quantization. Alternatively, Theis et al. [102] use the derivative of a simple identify function for the gradient of the quantization. Li et al. [103] employ this same trick to resolve their non-differentiable binarizer, while also utilizing an importance map to gauge pixel bitrate. Uniform additive noise has also been used as a proxy for the back propagation of the quantization step [104]. Lim et al. [105] explain how batch normalization layers in residual CNNs remove useful range information, and demonstrate how removing them increases quality, while also

reducing network size substantially.

If SAIs are treated independently, Dong et al. [106] propose a simple CNN for the super-resolution of images which they notably term Super-Resolution Convolutional Neural Network (SRCNN). The SRCNN is comprised of 3 CNN layers, strategically stacked such that they function as a feature extractor, non-linear mapping of said features, and a reconstuctor. They later go on to increase the speed of the network by a factor of 40, by adding a shrinkage layer after the feature detector, and work in low-resolution space until the final reconstruction [107]. In a similar attempt, the authors apply the network structure to the task of compression artifact removal, altering parts of the image instead of upscaling it [108]. Cavigelli, Hager, and Benini [109] are able to improve on these results by increasing the number of hidden layers, while also implementing metrics at multiple scales to their loss function. Ledig et al. [110] restructure the super-resolution task in the framework of GANs. By pairing a deep generator network made of 16 residual convolutional blocks, with a discriminator capable of identifying true high resolution images from generator ones, the coupled network is capable of creating images nearly indistinguishable from real ones.

4.3 Current Datasets

There are far less open source datasets available for light fields, than for more mainstream media forms such as images and videos. The Stanford Light Field Archive is the most well known, with the datasets often showing up in research papers [10]. The Stanford datasets consist of a mixture of different image resolutions. This can sometimes lead to issues, particularly in ML, as the images may need to be stretched

for a uniform aspect ratio throughout. This can lead to scaling issues along a dimension, for example in depth estimation where disparities will vary image to image. The Stanford Archives are divided into two groups. The Old Archives have light fields with relatively low spatial resolutions, whereas the New Archives only provide 13 light fields, but consist of 289 viewpoints each.

Another commonly used dataset in recent literature is the Flowers Dataset [3]. It is the largest light field dataset to date, made up of over 3300 light fields of flowers, taken with a Lytro Illum camera. At nearly 170GB, it is evident how problematic the large filesizes of light fields are. As the images are all of flowers, there is large correlation in terms of color distribution, visual features, and pixel depth between samples, which could cause overfitting and issues in generalization to other domains, when used as training data. Similarly, Lytro also offers some publicly available data captured on their cameras [111]. They share 25 high quality light fields, all in 1:1 image ratios, with a variety of viewpoints at close baselines, but require some formatting and preprocessing to access the data. Furthermore, with the company closing down, the associated file format is no longer generally supported.

MIT have their own achieve [112], with applications to other commonly cited papers [75]. This is a small dataset made up entirely of synthetically generated light fields, lacking complex textures.

The HCI Light Field Dataset as collected by groups from Heidelberg University and the University of Konstanz [113] is the most commonly used dataset for depth based solutions and depth estimation. The dataset consists of 24 light fields, each made up of 81 viewpoints. All the image slices are of equal size and aspect ratios. This is ideal for an input layer of a neural network, as the size would have to be

Dataset	Size	Type	Angular Resolution
Stanford New Archive	13	Real	17x17
Stanford Old Archive	6	Synthetic	32x32
Flowers Dataset	3343	Real	14x14
MIT Archive	5	Synthetic	5x5
HCI 4D Dataset	24	Synthetic	9x9
Lytro Archive	25	Real	10x10
LFSD [114]	100	Real	10x10
LCAV-31 [115]	31	Real	10x10
DDFF-12 [116]	720	Synthetic	9x9
Graz Dataset [117]	900	Synthetic	11x11
MPI Archive [118]	14	Mixed	101x1
V-SENSE [119]	11	Real	14x14

Table 4.1: Overview of all publicly available light field datasets.

fixed, while the aspect ratio assures our images would not suffer from any resizing artifacts. Furthermore, the data already comes segmented into a test/train split ideal for research purposes, and the true disparity maps are also provided if we choose to utilize them in the future. Each light field is made up of 9×9 densely packed viewpoints, each of 512×512 pixels along 3 color channels, in standard PNG format. The HCI Light Field Dataset is the inspiration for our dataset in Section 4.4, and will be used for testing purposes in Section 4.6.

Exact size and details for all publicly available light field datasets can be found in Table 4.1.

Despite all of the data sources outlined, the largest dataset consists of just 3300 light fields, and severely lacks scene diversity. As we aim to train a compressive neural network model, we will require far more data than is collectively available. In order to attain such a large data bank, we proceed to generate synthetic light fields using DR as detailed in the following section.

4.4 Data Generation

With a shortcoming in available data for the training of our ML model, we are forced to create our own. In order to produce a general model capable of compressing any light field, data diversity is a top priority in terms of lighting, textures, geometries, subjects, focal depths and camera baselines. Capturing such scenes by the thousands is a daunting task. Furthermore, commercial light field cameras are difficult to come by, while industrial cameras are expensive and require high levels of expertise to operate.

Instead, we proceed to create a synthetic dataset, largely using the methods and findings presented in Chapter 3 [120], while adopting the formats used in the HCI 4D Dataset [113]. We opt to use the Unity Game Engine, since the game engine is optimized to render scenes in a fraction of a second as oppose to Blender which can take minutes. We trade off the photorealism of Blender's ray tracing features for the speed improvements of Unity, due to the large amount of images we aim to produce.

We use the Classic Furniture Pack [1] house model from the Unity Asset Store as our backdrop for an environment, and iteratively spawn a light field camera at a random location, facing a random direction, inside the house. The house is made up of a collection of rooms featuring a variety of different characteristics, such as reflective surfaces, spot and point light sources, windows allowing external directional light, and a host of highly detailed typical home objects. We then load a random amount of the 200 available object models from the Garage Props Pack [2] in the camera's FoV to serve as occluders. We use this pack as we found the objects to contain a large

[1] `https://assetstore.unity.com/packages/3d/props/interior/classic-furniture-pack-54611`

[2] `https://assetstore.unity.com/packages/3d/props/tools-stuff-props-pack-151878`

Figure 4.1: Examples of objects used as occluders for each scene. Objects not to scale.

variety of shapes, sizes, albedos, specularities, and diffuse and normal surface maps. Examples of some occluders can be seen in Figure 4.1. We further randomize the position, and angular rotation of each occluder, and slightly perturb their size between each capture.

Next, the color and textures of all objects that lie in the viewing frustum are randomized using a collection of materials from Nobiax/Yughues [3], such as wood, copper, fur and fabrics, which can all be seen in Figure 4.2. The textures all include normal and reflectance maps, capable of producing non-Lambertian surface reflectance. To further iterate the light scattering, we alter the direction and intensity of the directional light in the scene, and the position and intensities of the point lights and area lights. Light color was indirectly perturbed via the diffusive properties of the environment surroundings. Reflection probes were also used in the scene to give mirrors and glass-like surfaces global reflectance, but were omitted from any

[3] `https://assetstore.unity.com/publishers/4986`

Parameter	Randomization Range	Unit
Camera Position X	$(0, 1)^{\dagger}$	
Camera Position Y	$(0, 1)^{\dagger}$	
Camera Position Z	$(0, 1)^{\dagger}$	
Camera Rotation (Yaw)	$[-180, 180)$	$^{\circ}$
Camera Rotation (Pitch)	$[-40, 40]$	$^{\circ}$
Camera Focal Length	$[40, 120]$	mm
Focus Distance	$[0.8, 2]$	m
Baseline	$[2, 10]$	mm
Color (Hue)	$[0, 1]$	
Color (Saturation)	$[0, 1]$	
Color (Value)	$[0, 1]$	
Textures	148	
Texture Tiling Scale	$[35, 65]$	%
Light Direction (Polar)	$[120, 180]$	$^{\circ}$
Light Direction (Azimuth)	$[0, 360)$	$^{\circ}$
Light Intensity	$[0, 100]$	%
Number of Occluders	$[1, 10]$	
Occluder Position (Distance)‡	$(0, 1]$	m
Occluder Position (Polar)‡	$[0, 22.5)$	$^{\circ}$
Occluder Position (Azimuth)‡	$[0, 360)$	$^{\circ}$
Occluder Rotation (Yaw)‡	$[-180, 180)$	$^{\circ}$
Occluder Rotation (Pitch)‡	$[-90, 90)$	$^{\circ}$
Occluder Rotation (Roll)‡	$[-180, 180)$	$^{\circ}$

Table 4.2: Parameter ranges used in the Domain Randomization.
† relative to environment bounds.
‡ relative to camera reference frame with optical axis indicating positive z-axis.

further alterations in the DR process. A full list of DR variable parameters, and their respective ranges is provided in Table 4.2.

A light field CA is built by aligning 9 rows and 9 columns of cameras in a grid, while keeping the optical axis of each parallel to one another, as was done by Honauer et al. Each camera is placed a baseline b horizontally and vertically from other adjacent cameras, which is allowed to vary from sample to sample and can be found in a supplementary *params.txt* file attached to each generated light field. However, the

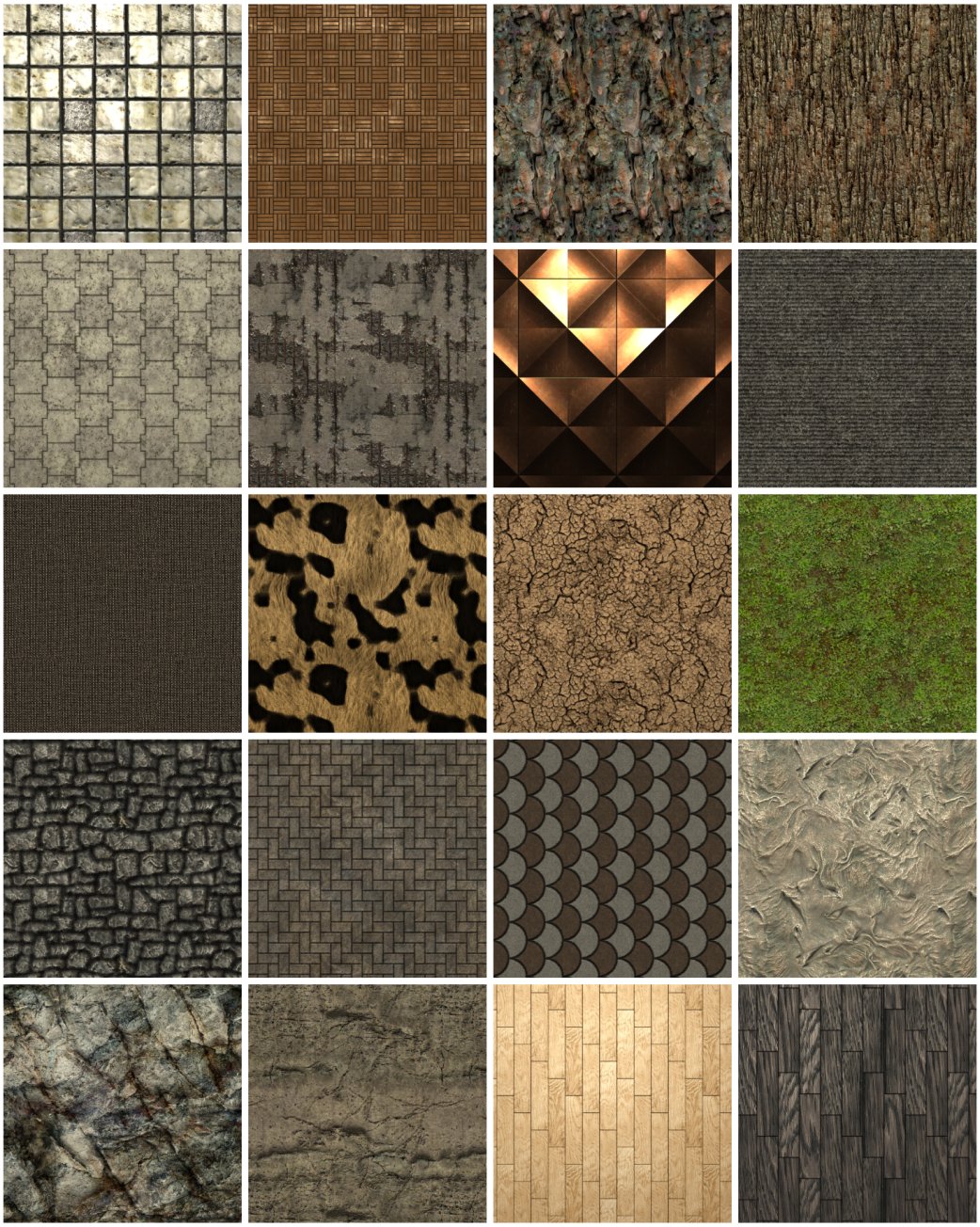


Figure 4.2: Examples of textures applied to scene objects.

sensor size of each camera is kept constant at 35 × 35 mm, producing a SAI of 512 × 512 pixels. With 81 SAIs, any larger resolutions would cause issues later on when attempting to train due to the memory constraints of GPUs. In order to align the view of each camera, we use a lens shift technique, typically produced using a tilt shift lens in real cameras. The lens shift is equivalent to sliding the sensor underneath the lens. Since our cameras do not take up any physical space in the game engine and are not visible, small baselines and large lens shifts don't suffer from the challenges that would be present in the real world. Similar to Honauer et al., our cameras are not rotated. A visual representation taken of the lens shift can be seen in Figure 4.3. It should be noted that though we randomize the focus distance and focal length each capture, all cameras in the array share these values. Using the lens shift, all the cameras are adjusted such that their viewports are aligned in the image plane, determined by the focus distance, with their optical axes perpendicular to the plane. Each camera's near and far clipping planes are set at .01 and 20 meters respectively.

We limit the DR to the physical characteristics of the environment and camera only. No post processing variability was applied to the camera renders, such as noise and gamma correction adjustments, which may be present under real imaging conditions. Blurring, vignetting and lens distortions effects are also not generally invertible, and careful attention must be paid as to correctly align the effects across SAIs. Furthermore, these effects take valuable time in the rendering step, causing substantial slowdowns. More often, this is implemented as a pre-processing step in the standard ML flow (as opposed to being baked in directly into the data), along transformations such as rotations, reflections, projections and cropping. A small amount of ambient occlusion is applied, which is technically applied as a post processing effect in game engines

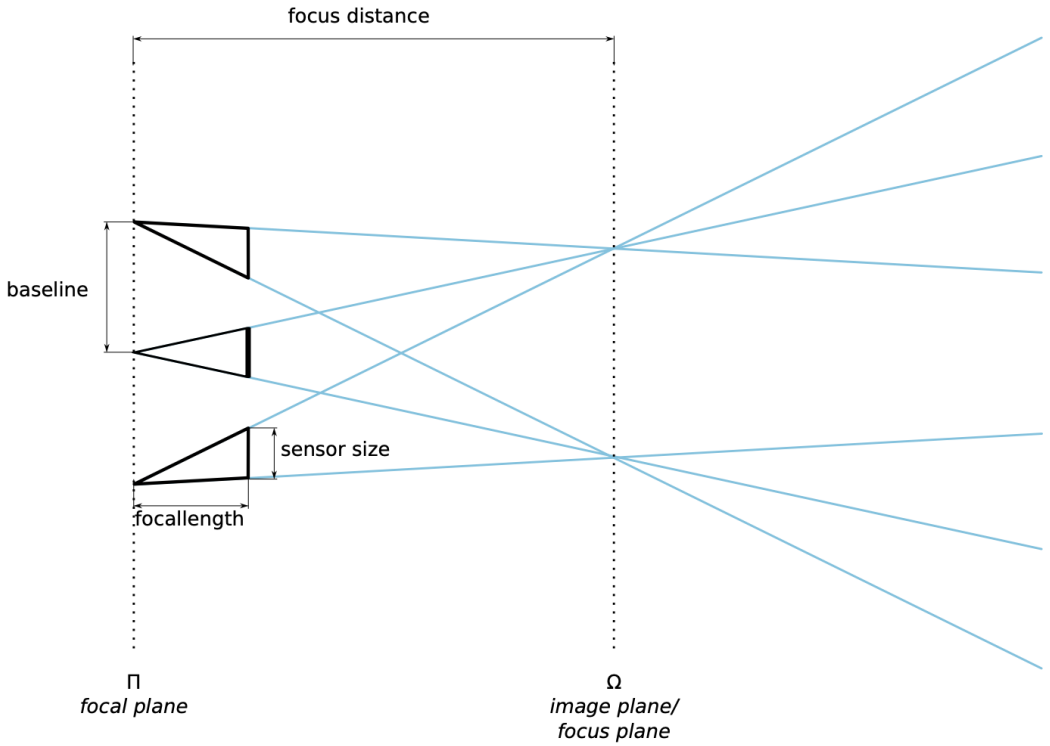


Figure 4.3: Lens shift of cameras in a light field CA, taken from the Supplementary Material of Honauer et al. [113]. Each camera is shifted such that the same cross sectional area on the image plane is in frame. The cameras are not rotated and their optical axes remain parallel.

due to the difficulty of approximating the true global scattering of light, but it is kept constant throughout. The addition of the ambient occlusion displayed negligible effects on rendering times.

Aside from the SAIs, the central camera of the CA captures a depth map and an object level segmentation maps. The depth map encodes the distance to the first collision a ray in a given pixel comes into contact with. This map is used in depth estimation and depth based image warping methods, and is also included in the HCI 4D Dataset. The segmentation map is a sparse image which assigns a unique color to each object visible in the viewport. This provides useful boundary information which when combined with the depth map can be strategically exploited for a variety of CV tasks.

The game engine is capable of generating a light field capture once every 5 seconds, including the time to save to disk. In between each capture, all DR parameters listed in 4.2 are randomized using a uniform sampling across the stated parameter ranges. Each new light field sample is made up of 81 independent view renderings, one from each camera, the depth and segmentation maps from the central view and the associated *params.txt* file for the CA parameters. The data synthesis was ran for 28 hours, generating 20,000 light field samples across a collection of diverse domains. The entire dataset takes up over 700 GB, with an average size of 35.8 MB per light field, which we refer to as the **LIAM-LF-Dataset** going forward. Examples of the generated images can be seen in Figure 4.4 and Appendix B.1.

Using a cloud storage provider, the data is uploaded using a direct Ethernet connection for further processing. Even with a direct connection and access to industrial level internet speeds, it takes over a day to upload the entire dataset, due

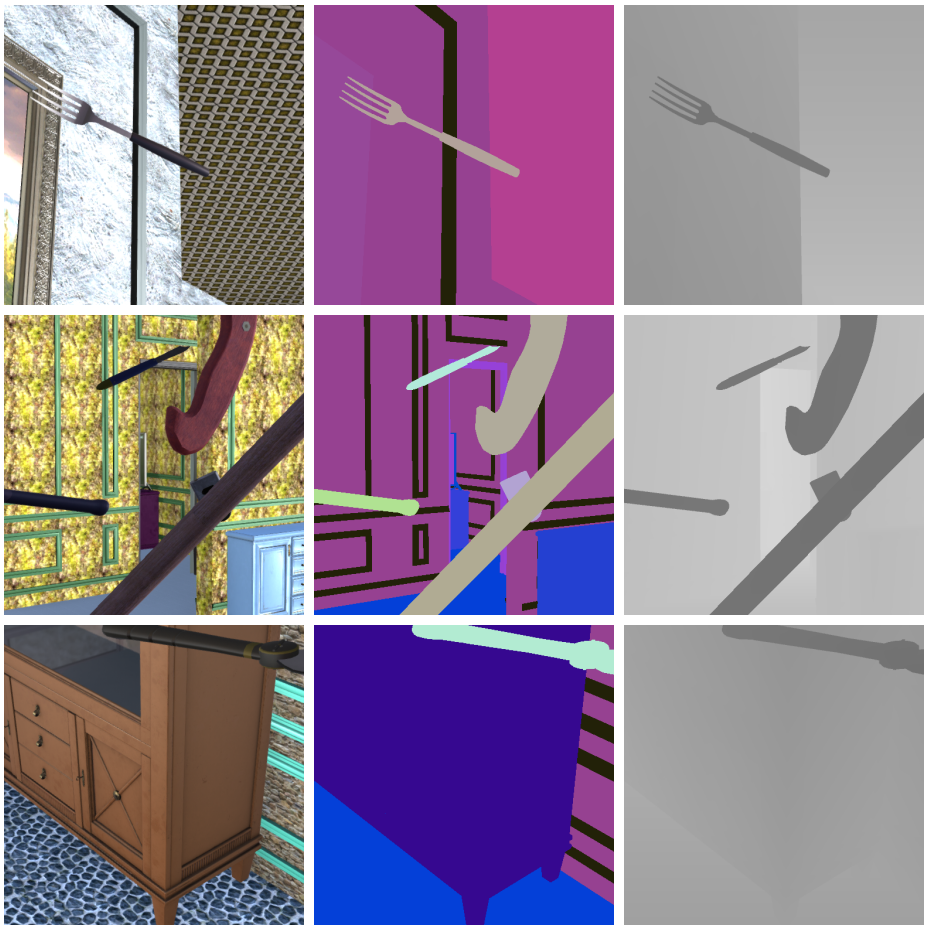


Figure 4.4: Examples of the central SAI of 3 synthetically generated light fields using DR. The associated segmentation and depth maps are also displayed.

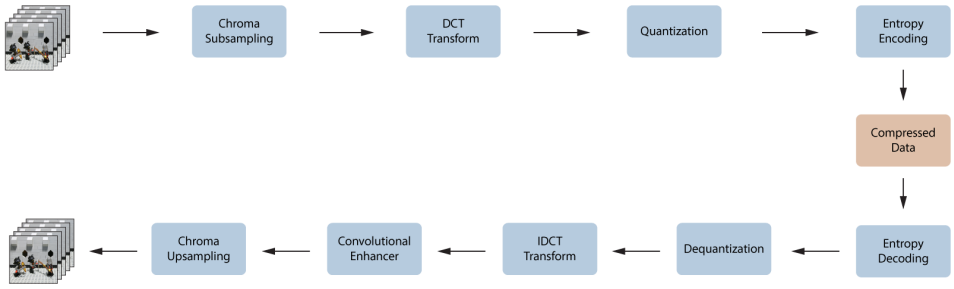


Figure 4.5: Compression and decompression branches of the proposed pipeline.

to its large size. Typical broadband limitations would cause wait times, orders of magnitude larger, making the necessity for file compression evident.

4.5 Compression Pipeline

Typically, a complete compression schema is comprised of a collection of smaller compartments, each making use of redundancies at different levels in order to optimally encode a signal. Our encoding scheme is a direct extension of the JPEG process to higher dimensions. Namely, we employ a standard chromatic subsampling technique, followed by a 4-Dimensional DCT of the light field. A quantizer is applied to the coefficients followed by an entropy encoder block at the end. The decoder slightly differs with the addition of the enhancement step. The encoding process is reversed for decoding the signal, with a convolutional enhancer applied to sharpen the images and remove any compression artifacts, before the final upsampling as shown in Figure 4.5.

4.5.1 Chroma Subsampling

Taking advantage of the human eye's greater sensitivity to luminance than chrominance, we begin by downsampling the red and blue chroma components. We first transform the initial gamma corrected SAIs from the RGB color space to the YCbCr color space. The transformation[4] is given by

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & 0.418688 & -0.08312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{4.1}$$

and reversed by

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.34436 & -0.714136 \\ 1 & 1.772 & 0 \end{bmatrix} \begin{bmatrix} Y \\ C_b - 128 \\ C_r - 128 \end{bmatrix} \tag{4.2}$$

where Y represents the luma and C_b and C_r are the blue and red chroma components respectively. The subsampling in this color space is commonly expressed as a ratio of the form N:a:b for a given patch made of 2 rows of pixels. N is the horizontal sampling of the luma, used for reference relative to the other 2 channels and is almost always set to 4. This can be thought of as the width of a patch in pixels. The a value represents the number of chromatic samples in the first row of N pixels, while b is a flag set to 0 if the second row of the patch carries down the same values from the top row, or a if it is sampled independently. The overall compression factor is calculated as

[4] `https://www.w3.org/Graphics/JPEG/jfif3.pdf`

$$\text{Compression Factor} = \frac{N + a + b}{12} \tag{4.3}$$

with the 4:4:4 ratio representing no subsampling and therefore no compression, while the 4:1:0 provides maximal subsampling with a compression factor of $\frac{5}{12}$ or a reduction of 58.3%.

Similar to JPEG and HEVC, we employ a chroma subsampling ratio of 4:2:0 in our pipeline, instantly cutting down our file size in half. Our results presented herein are based on the 4:2:0 standard, though alternative bitrates can be achieved utilizing the less aggressive subsampling schemes. For the sake of our pipeline, our variable bitrate will be entirely controlled by a quantization factor that we study further in Section 4.6.1.

It is important to note, that upsampling the chroma back is not a lossless process. The subsampling operation tosses away information that cannot be retrieved directly. Furthermore, while the color transformation across color spaces is invertible, there may be intermediate rounding errors introduced in the calculations which may lead to slight variations in the reconstructed images. These effects however are not on the order of being perceptually noticeable, and will be tolerable for our model.

4.5.2 4D DCT

In order to exploit maximal redundancies in our light field, we will work with the signal in the frequency domain. In this fashion, high frequencies are often just noise from the spatial domain, and can be toned down or entirely discarded similar to a typical low pass filter in signal processing. Moreover, patches in the light field are

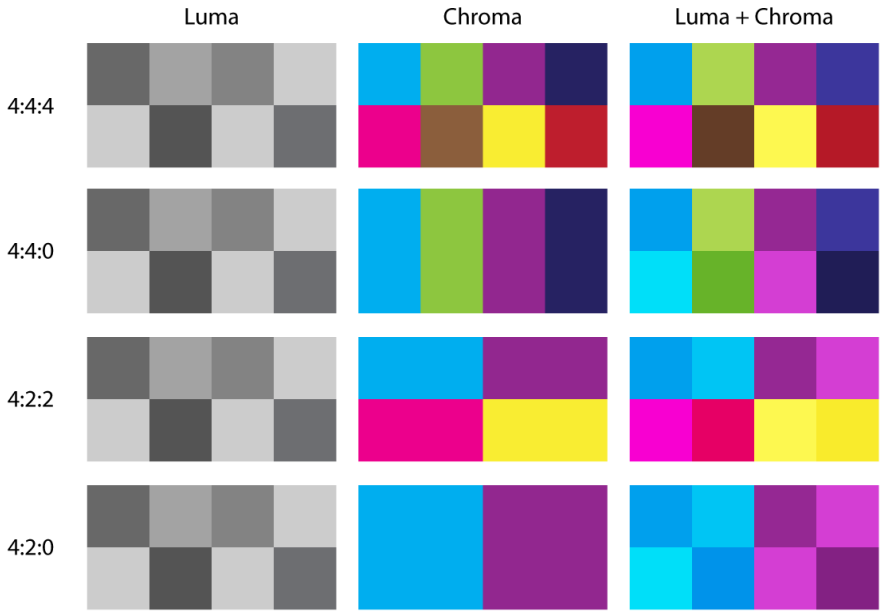


Figure 4.6: Examples of commonly used chroma subsampling ratios.

highly correlated and so we expect a handful of frequencies to be greatly representative of the entire area. Numerically, we can represent our discretized light field as the 5-dimensional tensor

$$L_{u,v,s,t,c} \qquad u, v = 1 \dots M, \quad s, t = 1 \dots N, \quad c = 1 \dots 3 \tag{4.4}$$

where u and v are the angular indices, s and t are the spatial indices and c is the indexed luma and chroma channels. The $M \times M \times N \times N \times 3$ light field tensor is subsampled as outlined in Section 4.5.1, and split it into 2 separate tensors; a luma tensor

$$L^{l}_{u,v,s,t,c} \qquad u, v = 1 \dots M, \quad s, t = 1 \dots N, \quad c = 1 \tag{4.5}$$

and a chroma tensor

$$L^{c}_{u,v,s,t,c} \qquad u,v = 1 \dots M, \quad s,t = 1 \dots N/2, \quad c = 1 \dots 2 \tag{4.6}$$

of size $M \times M \times N \times N \times 1$ and $M \times M \times \frac{N}{2} \times \frac{N}{2} \times 2$ respectively. In this form, each tensor is partitioned into blocks T of size $B_1 \times B_2 \times B_3 \times B_4$, and the DCT is applied across the u, v, s and t dimensions on each block using the 4-D DCT Type-II[5] given by

$$c_{i,j,k,l} = \sum_{u=1}^{B_1} \sum_{v=1}^{B_2} \sum_{s=1}^{B_3} \sum_{t=1}^{B_4} T_{u,v,s,t} \cos\left[\frac{\pi(i-1)}{B_1}\left(u - \frac{1}{2}\right)\right] \cos\left[\frac{\pi(j-1)}{B_2}\left(v - \frac{1}{2}\right)\right] \cos\left[\frac{\pi(k-1)}{B_3}\left(s - \frac{1}{2}\right)\right] \cos\left[\frac{\pi(l-1)}{B_4}\left(t - \frac{1}{2}\right)\right] \tag{4.7}$$

with $B_3, B_4 = 8$, where $T_{u,v,s,t}$ is the pixel value of a block at entry u, v, s, t, and $c_{i,j,k,l}$ is the associated DCT coefficient. The calculation are applied independently for each channel in the chroma scenario. The B_1 and B_2 terms are chosen to have spatial width and height of 8 pixels analogous to the patch sizes in JPEG encoding, while B_3 and B_4 will depend on the angular resolution of the data available. The block dimensions should be chosen as to be an integer factor of the tensor's dimension so we assume N is a multiple of 8. Each block is now represented by $64B_1B_2$ coefficients.

The DCT coefficients can be inverted back and patched into the original discretized light field tensors using the 4-D Inverse Discrete Cosine Transform (IDCT) Type-II[5]

[5]As the multidimensional DCT is simply the composition of one-dimensional DCTs, we extend the formulation to four dimensions directly from Rao and Yip [32]

$$T_{u,v,s,t} = \sum_{i=1}^{B_1}\sum_{j=1}^{B_2}\sum_{k=1}^{B_3}\sum_{l=1}^{B_4} c_{i,j,k,l} \cos\left[\frac{\pi(i-1)}{B_1}\left(u-\frac{1}{2}\right)\right]\cos\left[\frac{\pi(j-1)}{B_2}\left(v-\frac{1}{2}\right)\right] \cos\left[\frac{\pi(k-1)}{B_3}\left(s-\frac{1}{2}\right)\right]\cos\left[\frac{\pi(l-1)}{B_4}\left(t-\frac{1}{2}\right)\right] \quad (4.8)$$

4.5.3 Quantizer as a Convolutional Network

While transforming the light field into the frequency domain may lead to reductions in size depending on the distribution of the coefficients, this generally will not lead to large enough compression ratios. To make full use of the representation, we proceed to 1) remove any coefficients which have negligible effect on the image quality in a given batch, and 2) pack the coefficient as tightly as possible as to minimize the entropy of the data. To achieve this, a quantization tensor (similar to a quantization table) will be used to scale down (or scale up) each DCT coefficient, followed by the actual quantization operation which will round each entry to the nearest integer. In order to represent each coefficient with n bits, the quantization in effect normalizes the spectrum to the integer range $[0, 2^n - 1]$.

Generating a useful quantization tensor however largely depends on the characteristics of the light field data. We proceed to structure the task of building the quantization tensor, by transforming the quantization processes into a neural network, and training it on the LIAM-LF-Dataset in order to attain each entry.

The operations in each step of a typical quantization, as shown in Figure 4.7, can be represented with well behaved functions, aside from the rounding step. We transform the procedure into the paradigm of a neural network in Figure 4.8. The

Figure 4.7: A typical quantization process converts an image to it's DCT coefficients, normalizes the values using a quantization table and rounds the values. The encoding is imperfectly reconstructed by reversing the normalization and domain transform.

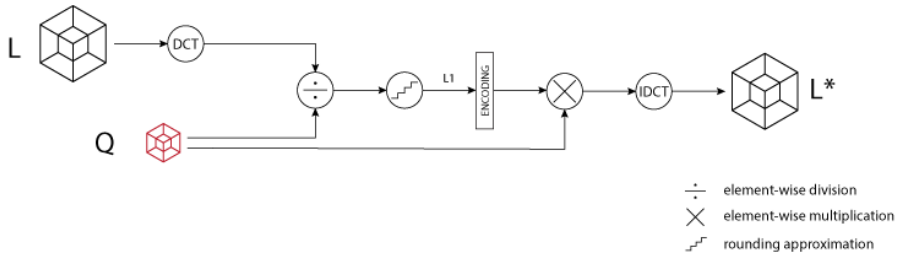


Figure 4.8: The quantization can be expressed as a quotient between the DCT coefficients of the light field L and the parameters of the quantization tensor Q, followed by an approximation to the round operations. By reversing the steps post quantization, the difference between the reconstructed light field L^* and L can be minimized, in order to solve for the optimal entries of Q.

DCT and IDCT can be described by a set of linear equations as given by 4.7 and 4.8, having smooth derivatives with respect to the input variables. This operation block can be trivially inserted into and backpropagated through, a neural network. The normalization steps, shown as $\otimes$ and $\oslash$ in Figure 4.8, are analogous to a single filter convolutional layer with kernel sizes of 1 in all dimensions and no bias term, adjusted such that the DCT coefficients are stacked in the typical channels dimension.

The rounding operations is not an invertible function, and hence information is lost in this step. Furthermore, the rounding function $f(x) = \lfloor x + \frac{1}{2} \rfloor$, is not differentiable and therefore gradients cannot be calculated during backpropagation. Instead, consider the function

$$s(x) = x - \left\lfloor x + \frac{1}{2} \right\rfloor \tag{4.9}$$

The function $s(x)$ is a shifted saw-tooth wave function commonly used in electrical engineering and signal processing, and can be decomposed into the Fourier series[6]

$$s(x) = \frac{1}{\pi} \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} \sin(2n\pi x) \tag{4.10}$$

which is still discontinuous at $x = n + \frac{1}{2}, n \in \mathbb{Z}$. Instead we can approximate $s(x)$, by

$$s(x) \approx s^*(x) = \frac{1}{\pi} \sum_{n=1}^{\infty} \frac{(\tau)^{n+1}}{n} \sin(2n\pi x) \tag{4.11}$$

where $s^*(x) \to s(x)$ as $\tau \to -1$, and is continuous and differentiable everywhere for $-1 < \tau < 0$. Using the Taylor expansion of the natural logarithm, we can get a closed

[6] `https://mathworld.wolfram.com/FourierSeriesSawtoothWave.html`

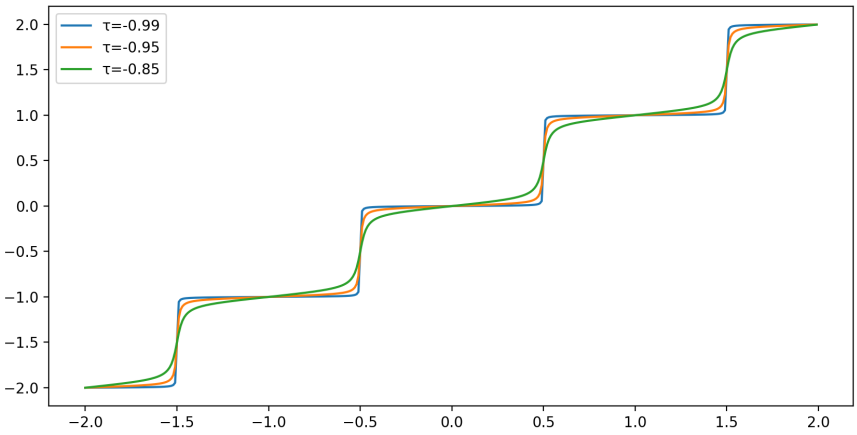


Figure 4.9: Effect of different τ values on $g(x)$.

form for Equation 4.11 arbitrarily close to $\tau = -1$, given by

$$s^*(x) = \frac{\tau}{\pi} \tan^{-1}\left(\frac{\tau \sin(2\pi x)}{1 - \tau \cos(2\pi x)}\right) \tag{4.12}$$

By rearranging, Equation 4.9, we approximate the rounding function $f(x)$ by

$$g(x) = x - \frac{\tau}{\pi} \tan^{-1}\left(\frac{\tau \sin(2\pi x)}{1 - \tau \cos(2\pi x)}\right) \tag{4.13}$$

where τ is a parameter that controls the tightness of the approximation, with values closer to -1 providing tighter fits. One can see how this parameter effects the slope of the function around integers values in Figure 4.9. Furthermore, the derivative of $g(x)$ is given by

$$g'(x) = 1 - \frac{2\tau^2(\cos(2\pi x)(1 - \tau\cos(2\pi x)) - \tau \sin^2(2\pi x))}{\tau^2 \sin^2(2\pi x) + (1 - \tau\cos(2\pi x))^2} \tag{4.14}$$

which is well defined everywhere. For full derivations of the closed form of $s*(x)$ and

the derivative of $g(x)$, see Appendix C. A τ value of -0.95 was chosen empirically such that $g'(x) > 0, \forall x$, while containing $\max(g'(x))$ as to allow the network to properly converge. The idea here is to approximate the optimal values of Q (learnable weights) for $f(x)$, by training the network using $g(x)$.

To train the network, we must incorporate a bitrate penalty into our loss function, otherwise minimizing only the distortion will lead to arbitrary small entries of Q, as to neutralize the effect of the quantization. We measure the distortion between the real and reconstructed light fields using the MSE and employ an L1 regularizer on the encoded signal. Since the probability distribution of the encoding (as measured by the frequency of each bit value) and hence its informational entropy, are discrete functions, they are non-differentiable with respect to their arguments, and so we can't incorporate it directly in our minimization. To circumvent this limitation, we enforce the L1 regularization on the encoding, which encourages sparsity [121] and therefore should lead to low entropy. Our loss function $\mathcal{L}$ can be written as

$$\mathcal{L}(L, w) = \underbrace{MSE(L, \mathcal{D}_w(\mathcal{E}_w(L)))}_{\text{distortion}} + \lambda \underbrace{\|\mathcal{E}_w(L)\|_1}_{\text{bitrate}} \tag{4.15}$$

where $\mathcal{E}_w$ and $\mathcal{D}_w$ are the encoder and decoder parts of the network, w are the weights, and λ is a coefficient controlling the desired tradeoff between the distortion and bitrate. The weights of the network are the entries of Q, and so the network is an optimization of the problem

$$Q = \hat{w} = \underset{w}{\operatorname{argmin}} \frac{1}{n} \left(\sum_{i=1}^{n} \mathcal{L}(L^{(i)}, w) \right) \tag{4.16}$$

over a dataset of n light field samples $L^{(i)}$.

Typically, compression algorithms allow for a variable bitrate, based on the task at hand. Adjusting λ is one such way to control the desired bitrate, with large values giving higher weight to the bitrate term in Equation 4.15, while lower values prioritize the distortion. However, training a new model for each bitrate, and storing each associated quantization tensor becomes problematic. Instead, we impose a quality factor q associated with Q, such that quantization at quality q is calculated as

$$L_q^* = IDCT\left(\left\lfloor \frac{DCT(L)}{qQ} + \frac{1}{2} \right\rfloor qQ\right) \tag{4.17}$$

where a value of $q = 1$ will give the initial bitrate associated with the trained λ value. Values of q between 0 and 1 produce higher perceptual quality reconstructions, while those above 1 will produce better bitrates. We explore this relation with greater detail in Section 4.6.1.

4.5.4 Entropy Encoding

Redundancy in the encoding $\mathcal{E}_w$ can further be exploited. As we mentioned above, the L1 regularization penalty should lead to a large proportion of the entries to be zero. This information can be tightly compressed losslessly, using entropy encoding techniques such as a Huffman encoding, or an arithmetic encoder. These techniques are capable of achieving compression rates near the theoretical limits defined by the Shannon entropy. The details of these methods will not be further described here, and we will report bpp values directly using Equation 2.5 under a logarithmic base 2, for theoretical purposes.

4.5.5 Enhancer Network Structure

Thus far, the compression process has largely mirrored that of a standard JPEG pipeline, extended to higher dimensions. To make full use of the power of modern deep learning and CNNs in particular, we propose a multi-resolution convolutional-based network which we call a Convolutional Neural Enhancer (CNE). Typical reconstructed images tend to display a number of common compression artifacts such as ringing, staircasing and blocking. Furthermore, loss of higher frequency details is often seen causing focal parts of the image to appear blurred. We outline the architecture and details of our CNE in what follows, and present our results in Section 4.6.2.

In the YCbCr color space, the majority of details and edges are controlled through the luma channel. Furthermore, perceptual metrics such as SSIM are highly dependant on the luminance of an image. As Dong et al. [106] and their results show, we only apply the luminance component of the light field through the CNE.

The CNE consists of convolutional blocks used to extract features at numerous different resolutions. Each convolutional block consists of a pair of 3D convolutional layers with Rectified Linear Unit (ReLU) activations. In order to effectively apply the convolution across the spatial dimensions, the angular dimensions u and v are initially collapsed together. The convolutions use a narrow kernel in order to encourage feature detection in a small neighbourhood of a given pixel at each resolution. In order to downsample intermediate representations, we employ a 3D maxpooling layer which approximately downsamples the resolution of the features in half. This causes subsequent layers to explore features at wider peripherals. The number of filters used in the convolutional blocks iteratively doubles, after each downsampling. After the

fifth block, the process is reversed, using a 3D transpose convolutional layer in order to upsample the low resolution features back up. A skip connection joins the output from the transposed layer with the input of the corresponding maxpooling layer at each resolution, as can been seen in Figure 4.10. The skip connections help guide the network in the early stages of training, encouraging the low resolutions to focus more on global features while the higher resolutions forward the local features directly to the end. Instead of a typical residual skip connection, we stack the tensors across the filters dimension as indicated by the $\textcircled{|}$ operator, such that subsequent blocks can independently make use of higher and lower level features as needed. A convolution is applied at the end to convert the features back into a single channel tensor. Finally, the compressed light field is added back to the output of the final convolutional block as indicated by the $\oplus$ operator, making the entire network a deep residual block. The network identifies the compression artifacts, and produces a residual such that when added to the initial compressed light field tensor, inverts the visual distortions.

We notably exclude the use of any normalization layers in our network. The normalization layers produce a substantial increase on the memory requirements of the network. This is not only a disadvantage at inference time, but also a hurdle during training due to hardware memory limits, which we discuss in Section 4.6.2. Furthermore, normalization layers remove the range flexibility of the images which is not trivially reversed without the incorporation of more intricate designs in the network. Our claim is supported by the findings of Lim et al. [105].

The CNE's job is to produce the best quality image, namely by the reduction of any artifacts present. As there are no bitrate restrictions, the loss function $\mathcal{L}$ is simply

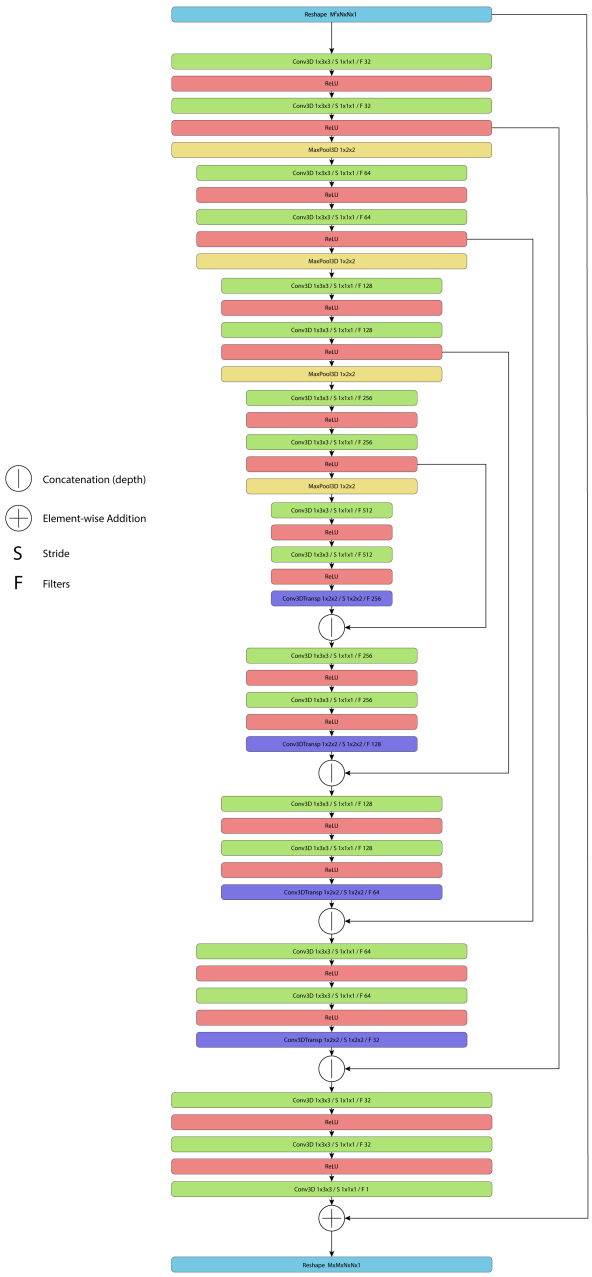


Figure 4.10: The network architecture resembles a deep residual block, made up of a sequence of convolutional blocks at different resolutions. Concatenations are done across the filters dimension, while the addition is performed element-wise.

$$\mathcal{L}(L, L^*, w) = MSE(L, \mathcal{N}_w(L^*)) \tag{4.18}$$

where L and L^* are the clean and noisy light fields respectively, $\mathcal{N}_w$ is the output of the neural enhancer, and w are the network's weights.

4.6 Experimentation

4.6.1 Neural Quantizer

Training

Utilizing the architecture described in Section 4.5.3, we proceed to train 2 separate models; one for the luma channel and one for the chroma channels. We present results that show a substantial increase in quality for a given bitrate vs standard image compression techniques on a handful of evaluation metrics. Furthermore, we provide a precise function for the selection of the quality parameter q, dependant on the desired bitrate.

Using the LIAM-LF-Dataset, we first preprocess each SAI by first converting the image into the YCbCr color space, and split it along the channels. As we outlined in Section 4.5.1, the chromatic channels are subsampled using a 4:2:0 sampling ratio, effectively reducing the resolution of the blue and red chrominance channels in half. We construct a $9 \times 9 \times 512 \times 512 \times 1$ tensor to represent the luma of a given light field sample, and feed it into our network. The 20,000 sample dataset is split into 16,000 train and 4,000 validation light fields, following a standard 80-20 heuristic.

We set $B_1, B_2 = 9$, as to span the entire angular resolution of the light fields,

giving us a quantization tensor of size $9 \times 9 \times 8 \times 8 \times 1$.

As each light field sample is equivalent to 81 standard images, we are limited on the size of each batch, due to memory constraints on the GPU. We proceed to train the network with a batch size of 1, and $\lambda = 10^{-4}$ which we found empirically to produce a good tradeoff between the rate and distortion terms. The model is trained for 2 epochs, with an initial learning rate of 10^{-2}, and decreased to 10^{-3} halfway through. Surprisingly, the model converges rather quick, with stagnation in the loss after only 4,000 iterations. We note that both the Adam and RMSProp optimizers displayed similar results. The training time for the whole process was 26 hours on a NVIDIA Tesla T4.

The training was repeated independently for the chroma, with 2 slight differences. First, since the chroma was subsampled, the tensor's spatial dimensions are adjusted to 256×256. Second, since the same quantization tensor will be used for both chromatic channels, we randomly sample just one of the 2 channels each iteration at train time.

Quality vs Bitrate

The quality parameter q does not correlate linearly to a perceptual metric during quantization. Furthermore, the degree of compression is often limited by bandwidth constraints. Therefore, it is useful to study how varying q effects the overall bitrate of a compressed sample. To do this, we evaluate the model on the validation set, across a multitude of q values. The results are summarized in Figure 4.11. The average bitrate appears to be modelled by a smooth decaying function. Assuming a general exponential function of the form

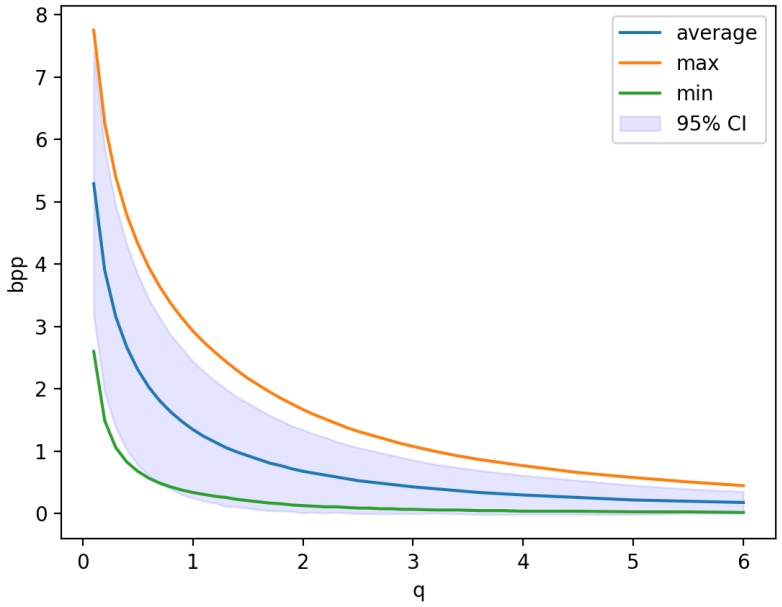


Figure 4.11: Bitrates for a given quality q, as measured on the validation set. The blue region represents the average +/- 1.96 standard deviations, while the green and yellow indicate the best and worst compression samples.

$$b(q) = \alpha e^{-\beta q^{\gamma}} \tag{4.19}$$

we approximate a relation between two. Using the Generalized Reduced Gradient (GRG) [122] non-linear programming solver, we find the values

$$\alpha = 16.158$$

$$\beta = 2.47$$

$$\gamma = 0.346$$

minimize the MSE. The function $b(q)$ gives us the average bits used per pixel for a

given quality factor q for our 4-D Quantizer. However, as we mentioned previously, typically we want the inverse relation. Inverting the function, we get

$$q(b) = \left(\frac{\ln(b) - 2.782}{-2.47} \right)^{2.89} \tag{4.20}$$

where $q(b)$ is the best estimate for the quality factor in the compression, for a desired bitrate b. Depending on the task at hand, Equation 4.20 provides a reasonable approximation for the value of q required. However, it is to be noted that this relation is based on average bitrate, and may vary greatly sample to sample. Nevertheless, it is a useful heuristic, and becomes more accurate as the number of samples increases. Alternatively, one can use a more conservative approximation using the worst case estimates as shown by the maximal curve in Figure 4.11, leading to the alternative set of parameters

$$\alpha_{max} = 13.037$$

$$\beta_{max} = 1.50$$

$$\gamma_{max} = .453$$

for a more stringent limit on the bitrate needed.

Results

We proceed to compare the model against other modern image compression algorithms. For evaluation purposes, we test the model on the HCI 4D Test Dataset. Since our model is trained entirely on the LIAM-LF-Dataset, this has the added advantage of

evaluating the quality of the DR used to generate our synthetic dataset. For each scene we calculate the PSNR, SSIM, MS-SSIM and LPIPS associated with each SAI after conversion back to the RGB color space, and present the averages across the entire light field in Figures 4.12, 4.13, 4.14 and 4.15. As the results show, our 4D Quantizer outperforms the other methods substantially in all metrics. Notably, our quantizer produces much more stable similarity and perceptual metrics across varying bitrates. Our method only underperforms the JPEG2000 standard for the *Origami* scene in terms of PSNR, for large bitrates. However, for bitrates below 0.4 bpp, our method remains optimal. Despite the model being trained to only minimize the MSE, all perceptual measures are largely improved.

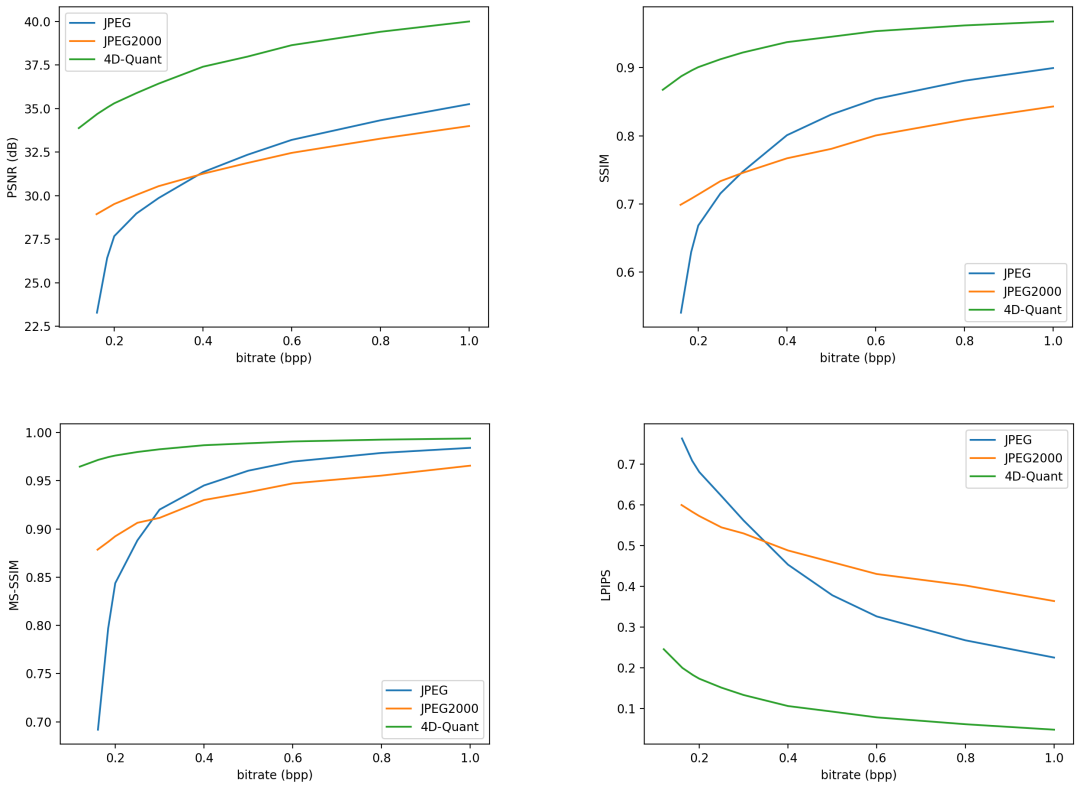


Figure 4.12: Comparison of PSNR, SSIM, MS-SSIM and LPIPS scores on the *Bedroom* scene from the HCI 4D Test Dataset. For the JPEG and JPEG2000 methods, each SAI is compressed individually and scores are averaged out across the entire light field.

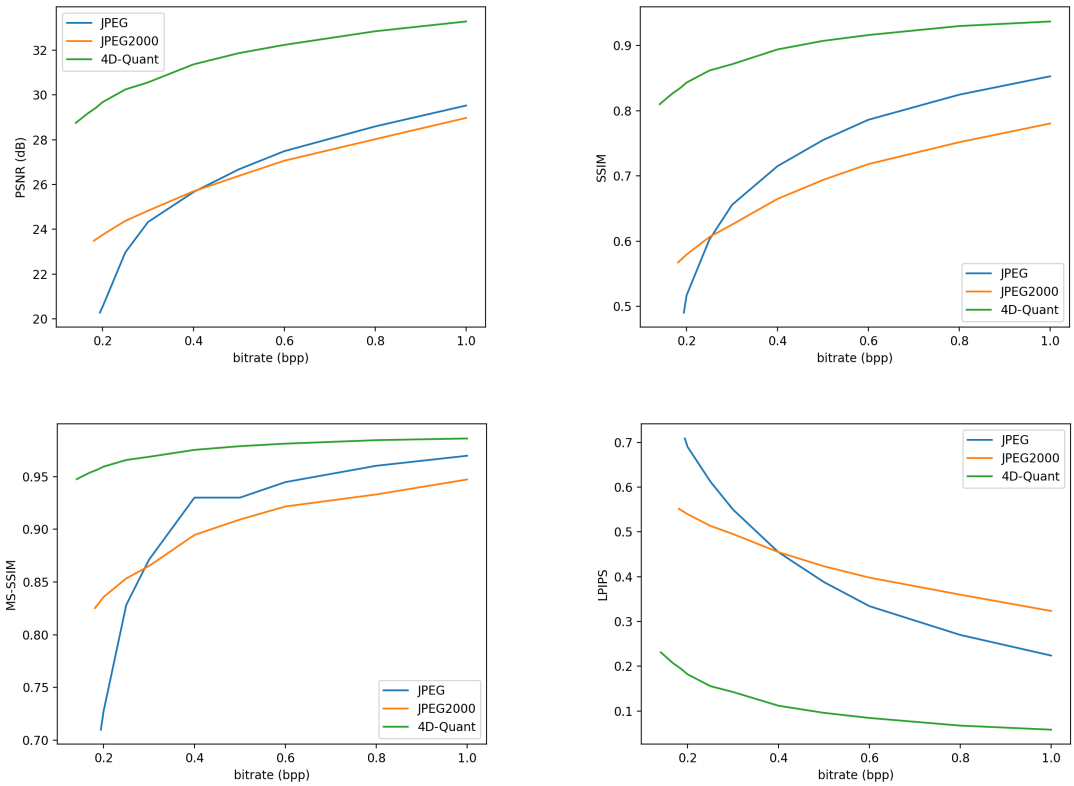


Figure 4.13: Comparison of PSNR, SSIM, MS-SSIM and LPIPS scores on the *Bicycle* scene from the HCI 4D Test Dataset.

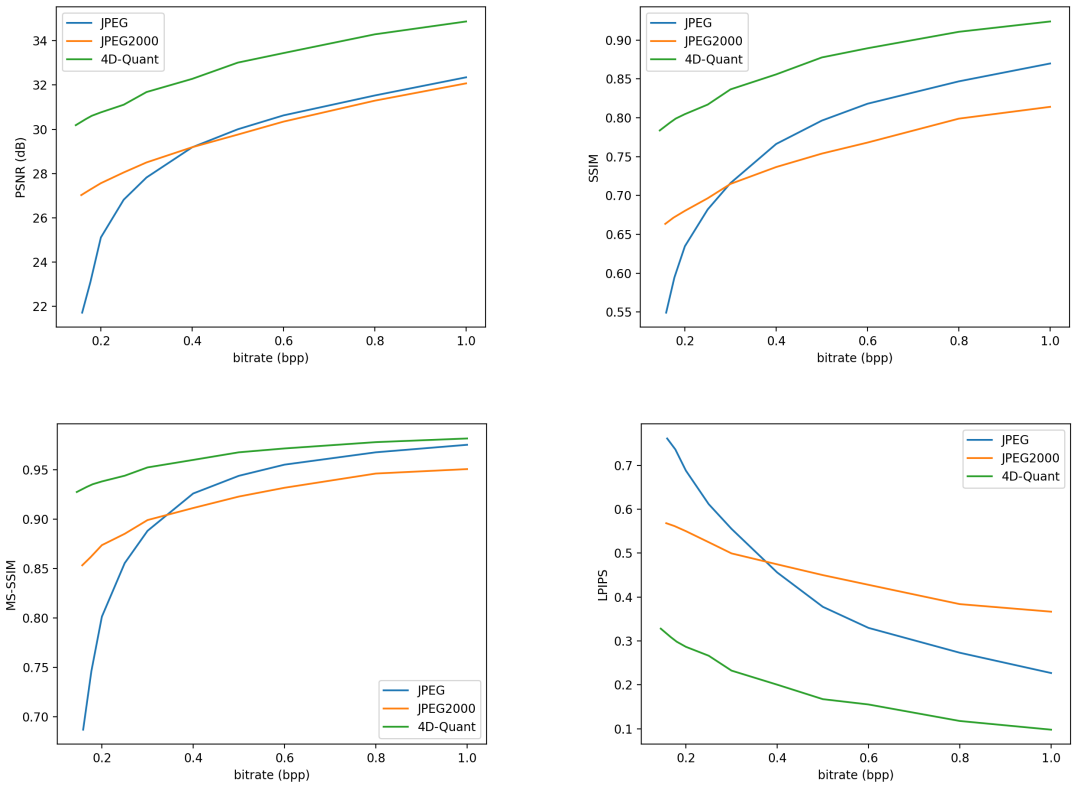


Figure 4.14: Comparison of PSNR, SSIM, MS-SSIM and LPIPS scores on the *Herbs* scene from the HCI 4D Test Dataset.

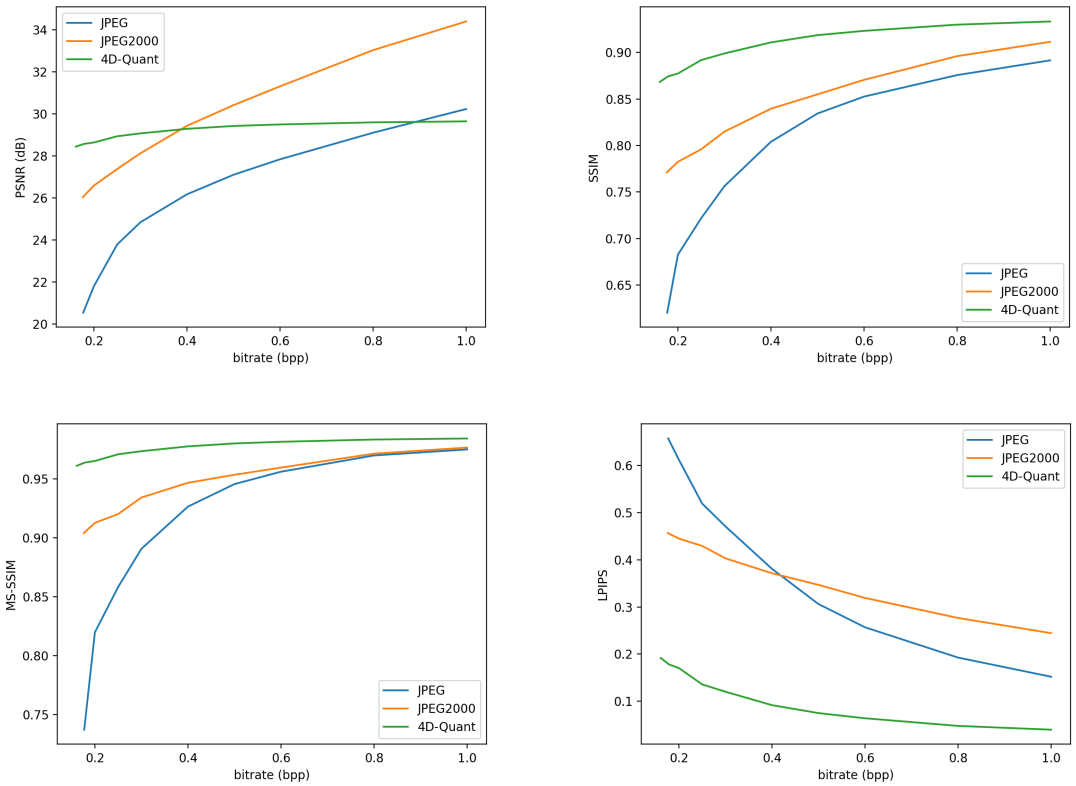


Figure 4.15: Comparison of PSNR, SSIM, MS-SSIM and LPIPS scores on the *Origami* scene from the HCI 4D Test Dataset.

Interestingly, the results of the 4D Quantizer at low bitrates are comparable to those of JPEG and JPEG2000 at bitrates 5-10 times higher. The diminishing degradation of our model at bitrates below 0.2 bpp is particularly noteworthy, as this is likely the realistic range needed in practice to share and stream light fields. Furthermore, the DR produces large enough variability for the model to bridge the gap between domains. This is evident by the high evaluation metrics on data from a distribution the model has never seen before. We will explore these ultra compressed bitrates further in Section 4.6.2.

4.6.2 Enhancer

Training

As our CNE is paired particularly to the 4D Quantizer presented in the previous section, we need to preprocess our dataset accordingly. The CNE itself sits on top of the quantizer, and can be interpreted as a separate block. We strip the luma channel of the reconstructed light field L^*, and feed it into our network. As L^* depends on the quality factor q, naturally a separate model needs to be trained for each q. However, we find that the model remains resilient across all levels, when trained on a single, moderately aggressive quantization. Practically this is desirable as only a single model is required for decompression. We find that training the network on lower quality factors fails to generate significant enough artifacts for generalizing to higher compression factors. Conversely, training on an overly aggressive q causes severe blurring when transferred to lightly compressed samples. While all values greater than 0 are meaningful depending on the target required, we find setting $q = 3$ to generally produce a fine balance between artifact prioritization and compression generalization. Our findings are in line with the results presented by Cavigelli, Hager, and Benini [109].

The large memory required to build the network, along with the time consumption of building each light field and transforming it to the frequency domain and back, makes training the model challenging. In particular, the network becomes too large to store on the GPU, when the entire light field is passed. To deal with this, we slice each light field into hyperblocks of size $9 \times 9 \times 128 \times 128 \times 1$, and train the network in batches of 3. We train the network for 4 epochs using the same train-validation

split from Section 4.6.1 using the Adam optimizer. The associated learning rate was initially set to 10^{-2} and lowered to 10^{-3}, 10^{-4} and 10^{-5} at the end of each epoch, respectively, in order to encourage finer optimization of the network weights. Training took 4 days on a NVIDIA Tesla T4.

Results

We build on the results from the previous section, by incorporating the combination of the 4D Quantizer with the CNE. Detailed results are presented in Table 4.3, across the HCI 4D Test Dataset, at a variety of different bitrates. We generate the full light fields iteratively, one hyperblock at a time. All metrics were calculated in the RGB color space, after upsampling in the chroma channels. Values were interpolated where needed to align with specific bitrates.

The *Bedroom*, *Bicycle* and *Herbs* scene all display a similar pattern, where our quantization tensor leads to optimal results across PSNR, SSIM and MS-SSIM metrics for higher bitrates, until an inflection point when the CNE overtakes it. This point is likely related to the quality factor chosen during training of the CNE. By lowering the q chosen at train time, we can reach this inflection point at higher bitrates. This unfortunately, comes at the cost of less significant improvements at lower bitrates. For bitrates greater than 0.40 bpp, JPEG2000 produces best PSNR results on the *Origami* scene. Our models, again outperform the standard compression algorithms at rates below that.

In terms of SSIM and MS-SSIM, our models produce substantially higher perceptual scores across the board, with the CNE again performing best at ultra high levels of compression. Surprisingly, the LPIPS measure suggests that the addition of the

CNE actually leads to less favourable results. The LPIPS score is calculated based on a CNN trained on real images, and perhaps it is able to identify and penalize irregularities generated from our enhancer. Alternatively, it may have been trained on images which may have undergone Fourier based compression, leading to a bias towards the very artifacts we aim to reduce. For a visual comparison, we include a SAI from the *Origami* light field using all the compression methods, at approximately the same bitrate, and a closeup of the edge of the bowl in Figures 4.16 and 4.17. Notice, particularly at the lowest rates, the slight ringing artifacts on the lip of the bowl, and the blocking present throughout. Our CNE reduces these artifacts greatly, at the expense of a slight drop in image sharpness.

Figure 4.16: Visual comparison of the top left SAI of the *Origami* light field from the HCI 4D Dataset. Bitrate values reported are accurate to within 0.005 bits.

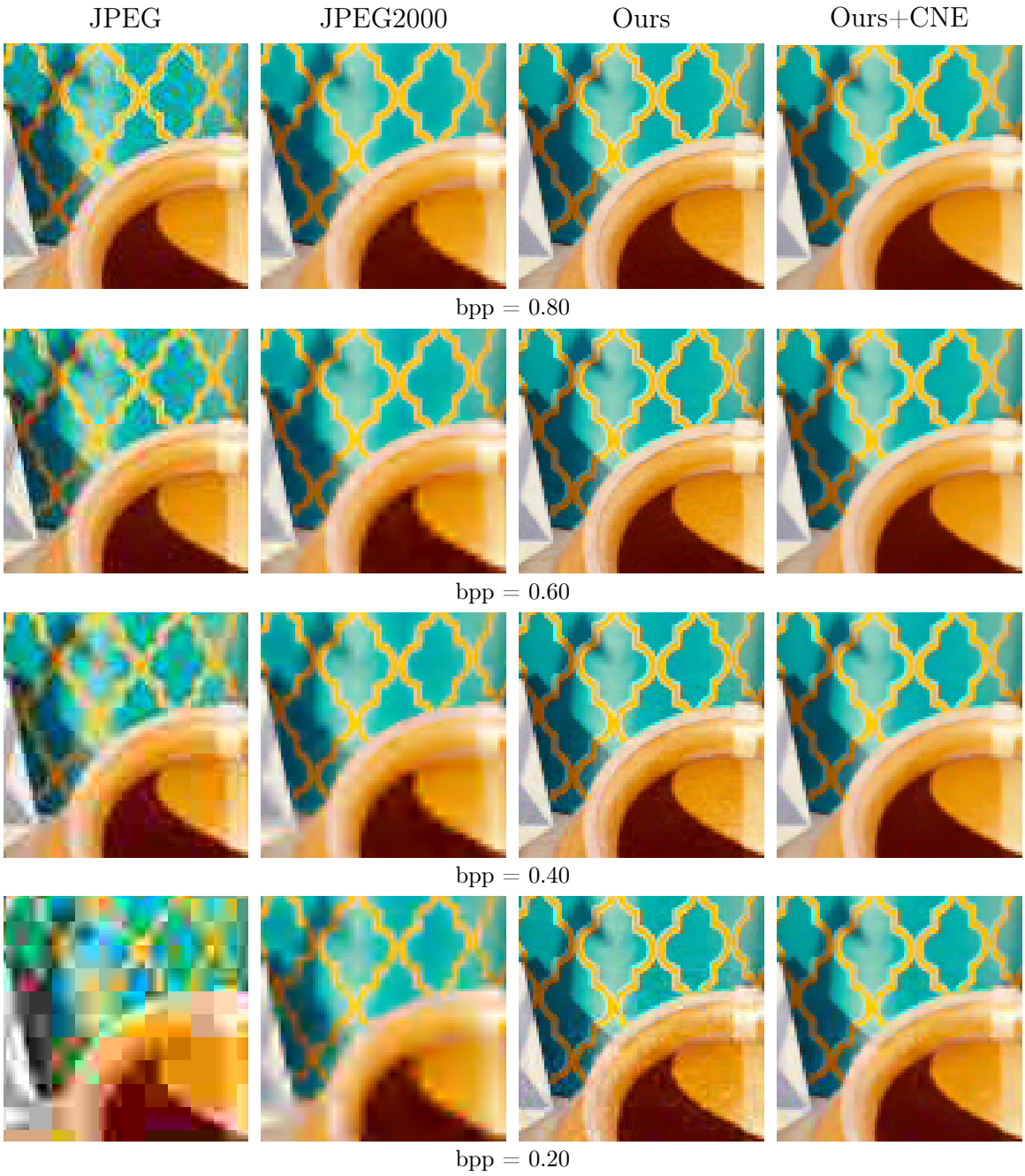


Figure 4.17: Visual comparison of an enlarged region from the *Origami* scene. While the JPEG quality quickly degrades, and the JPEG2000 appears to blur the image, our quantizer is capable of maintaining fine details even at low bitrate. Our enhancer further removes ringing and blocking artifacts as seen on the lip of the bowl. Bitrate values reported are accurate to within 0.005 bits.

Table 4.3: Detailed Results for the HCI 4D Test Dataset

		PSNR				SSIM				MS-SSIM				LPIPS			
Scene	bpp	JPEG	JPEG2000	Ours	Ours + CNE	JPEG	JPEG2000	Ours	Ours + CNE	JPEG	JPEG2000	Ours	Ours + CNE	JPEG	JPEG2000	Ours	Ours + CNE
bedroom	1.00	35.251	33.993	**39.997**	36.992	0.8995	0.8431	**0.9678**	0.9245	0.9841	0.9656	**0.9938**	0.9841	0.2249	0.3637	**0.0477**	0.1803
	0.80	34.322	33.268	**39.405**	36.848	0.8809	0.8239	**0.9620**	0.9220	0.9788	0.9553	**0.9926**	0.9835	0.2673	0.4021	**0.0611**	0.1857
	0.60	33.198	32.452	**38.63**	36.627	0.8541	0.8006	**0.9536**	0.9181	0.9698	0.9472	**0.9907**	0.9825	0.3260	0.4302	**0.0780**	0.1940
	0.50	32.340	31.870	**37.977**	36.411	0.8314	0.7810	**0.9454**	0.9143	0.9604	0.9381	**0.9888**	0.9814	0.3780	0.4592	**0.0921**	0.2015
	0.40	31.344	31.254	**37.399**	36.193	0.8010	0.7671	**0.9375**	0.9102	0.9451	0.9300	**0.9868**	0.9802	0.4533	0.4881	**0.1058**	0.2100
	0.30	29.858	30.540	**36.428**	35.751	0.7474	0.7457	**0.9222**	0.9017	0.9201	0.9115	**0.9826**	0.9776	0.5617	0.5299	**0.1329**	0.2271
	0.25	28.970	30.039	**35.884**	35.472	0.7156	0.7336	**0.9124**	0.8960	0.8882	0.9064	**0.9798**	0.9757	0.6218	0.5448	**0.1510**	0.2400
	0.20	27.670	29.515	**35.299**	35.140	0.6685	0.7140	**0.9008**	0.8890	0.8438	0.8924	**0.9761**	0.9734	0.6809	0.5727	**0.1730**	0.2546
	0.18	26.414	29.281	**35.086**	35.007	0.6295	0.7078	**0.8963**	0.8861	0.7968	0.8866	**0.9747**	0.9723	0.7086	0.5832	**0.1818**	0.2600
	0.16	23.276	28.930	34.692	**34.757**	0.5407	0.6989	**0.8876**	0.8805	0.6919	0.8785	**0.9717**	0.9703	0.7632	0.5998	**0.1995**	0.2726
	0.12	-	-	33.873	**34.194**	-	-	**0.8676**	0.8671	-	-	0.9646	**0.9654**	-	-	**0.2454**	0.3014
bicycle	1.00	29.529	28.975	**33.285**	32.098	0.8528	0.7804	**0.9369**	0.9051	0.9697	0.9472	**0.9861**	0.9791	0.2238	0.3234	**0.0582**	0.1249
	0.80	28.596	28.025	**32.848**	31.912	0.8246	0.7517	**0.9299**	0.9005	0.9602	0.9330	**0.9845**	0.9779	0.2699	0.3598	**0.0673**	0.1312
	0.60	27.490	27.063	**32.239**	31.618	0.7861	0.7181	**0.9161**	0.8930	0.9447	0.9216	**0.9812**	0.9758	0.3342	0.3979	**0.0844**	0.1414
	0.50	26.678	26.385	**31.871**	31.421	0.7549	0.6937	**0.9071**	0.8877	0.9300	0.9092	**0.9788**	0.9742	0.3880	0.4233	**0.0958**	0.1485
	0.40	25.661	25.694	**31.368**	31.136	0.7149	0.6647	**0.8940**	0.8798	0.9300	0.8946	**0.9753**	0.9719	0.4546	0.4548	**0.1117**	0.1587
	0.30	24.323	24.828	30.563	**30.626**	0.6556	0.6253	**0.8713**	0.8649	0.8711	0.8651	**0.9687**	0.9672	0.5499	0.4952	**0.1424**	0.1803
	0.25	22.964	24.374	30.246	**30.402**	0.6020	0.6062	**0.8616**	0.8580	0.8283	0.8534	**0.9657**	0.9649	0.6127	0.5136	**0.1555**	0.1901
	0.20	20.533	23.751	29.680	**29.991**	0.5168	0.5793	0.8432	**0.8447**	0.7274	0.8357	0.9594	**0.9603**	0.6900	0.5396	**0.1819**	0.2085
	0.18	20.272	23.487	29.422	**29.790**	0.4902	0.5673	0.8343	**0.8380**	0.7099	0.8252	0.9563	**0.9579**	0.7087	0.5516	**0.1946**	0.2185
	0.16	-	-	29.186	**29.601**	-	-	0.8260	**0.8316**	-	-	0.9534	**0.9556**	-	-	**0.2067**	0.2272
	0.14	-	-	28.751	**29.251**	-	-	0.810	**0.8196**	-	-	0.9475	**0.9511**	-	-	**0.2311**	0.2442
herbs	1.00	32.349	32.075	**34.872**	34.126	0.8699	0.8141	**0.9240**	0.9020	0.9754	0.9508	**0.9818**	0.9763	0.2269	0.3666	**0.0979**	0.1488
	0.80	31.528	31.293	**34.291**	33.848	0.8469	0.7989	**0.9107**	0.8951	0.9678	0.9463	**0.9781**	0.9742	0.2731	0.3838	**0.1177**	0.1603
	0.60	30.628	30.346	**33.444**	33.356	0.8181	0.7681	**0.8895**	0.8824	0.9553	0.9319	**0.9717**	0.9700	0.3297	0.4278	**0.1553**	0.1825
	0.50	30.001	29.761	33.013	**33.076**	0.7964	0.7539	**0.8777**	0.8746	0.9440	0.9230	**0.9678**	0.9674	0.3777	0.4499	**0.1674**	0.1956
	0.40	29.186	29.189	32.278	**32.539**	0.7663	0.7366	0.8559	**0.8589**	0.9261	0.9114	0.9601	**0.9616**	0.4559	0.4744	**0.2004**	0.2227
	0.30	27.822	28.502	31.683	**32.061**	0.7160	0.7149	0.8366	**0.8440**	0.8883	0.8992	0.9525	**0.9558**	0.5552	0.4993	**0.2324**	0.2491
	0.25	26.817	28.049	31.110	**31.561**	0.6820	0.6964	0.8170	**0.8278**	0.8556	0.8853	0.9441	**0.9490**	0.6115	0.5248	**0.2663**	0.2785
	0.20	25.111	27.564	30.765	**31.246**	0.6346	0.6802	0.8047	**0.8178**	0.8014	0.8738	0.9383	**0.9441**	0.6879	0.5500	**0.2866**	0.2975
	0.18	23.090	27.255	30.609	**31.105**	0.5943	0.6714	0.7990	**0.8126**	0.7462	0.8615	0.9355	**0.9418**	0.7360	0.5616	**0.2982**	0.3072
	0.16	21.712	27.025	30.461	**30.964**	0.5491	0.6634	0.7936	**0.8078**	0.6870	0.8534	0.9328	**0.9395**	0.7610	0.5681	**0.3086**	0.3151
	0.14	-	-	30.185	**30.703**	-	-	0.7837	**0.7993**	-	-	0.9276	**0.9351**	-	-	**0.3280**	0.3310
origami	1.00	30.223	**34.388**	29.638	29.335	0.8916	0.9115	**0.9334**	0.9151	0.9751	0.9767	**0.9844**	0.9796	0.1521	0.2445	**0.0398**	0.1162
	0.80	29.097	**33.025**	29.590	29.308	0.8757	0.8961	**0.9300**	0.9133	0.9700	0.9715	**0.9835**	0.9791	0.1927	0.2767	**0.0478**	0.1200
	0.60	27.833	**31.304**	29.491	29.249	0.8526	0.8706	**0.9233**	0.9097	0.9562	0.9597	**0.9816**	0.9777	0.2569	0.3192	**0.0640**	0.1265
	0.50	27.096	**30.407**	29.418	29.207	0.8344	0.8550	**0.9187**	0.9072	0.9457	0.9536	**0.9802**	0.9769	0.3062	0.3469	**0.0750**	0.1317
	0.40	26.168	**29.425**	29.283	29.124	0.8040	0.8396	**0.9109**	0.9029	0.9266	0.9468	**0.9777**	0.9752	0.3809	0.3715	**0.0918**	0.1397
	0.30	24.839	28.119	**29.065**	28.984	0.7563	0.8149	**0.8990**	0.8960	0.8908	0.9343	**0.9736**	0.9725	0.4709	0.4031	**0.1203**	0.1556
	0.25	23.782	27.369	**28.929**	28.893	0.7218	0.7959	**0.8919**	0.8918	0.8584	0.9203	**0.9710**	0.9707	0.5192	0.4294	**0.1358**	0.1635
	0.20	21.799	26.584	28.633	**28.689**	0.6826	0.7825	0.8774	**0.8829**	0.8200	0.9128	0.9653	**0.9667**	0.6121	0.4450	**0.1705**	0.1850
	0.18	20.538	26.031	28.567	**28.641**	0.6202	0.7710	0.8744	**0.8809**	0.7373	0.9041	0.9640	**0.9657**	0.6573	0.4568	**0.1780**	0.1895
	0.16	-	-	28.435	**28.544**	-	-	0.8684	**0.8771**	-	-	0.9613	**0.9638**	-	-	**0.1918**	0.1983

We further analyze the quality differences between our models, particularly at the lower bitrates, on the LIAM-LF-Dataset. Figure 4.18 shows the average metrics across the validation fold of the synthetic light fields. On average, the CNE improves performance for all rates, across all metrics, including LPIPS. These results contradict our earlier findings on the HCI 4D Dataset, indicating an existence of a slight domain gap between the two sets. All four metrics were on average lower than their corresponding scores on the HCI 4D Dataset. This is likely due to overly packed, heavily textured scenes in the LIAM-LF-Dataset. Our results are therefore likely to underestimate the reconstruction quality of natural captures, and require further analysis on a real-world dataset for better estimates in practice. Further examples of our CNE can be seen in Appendix D.

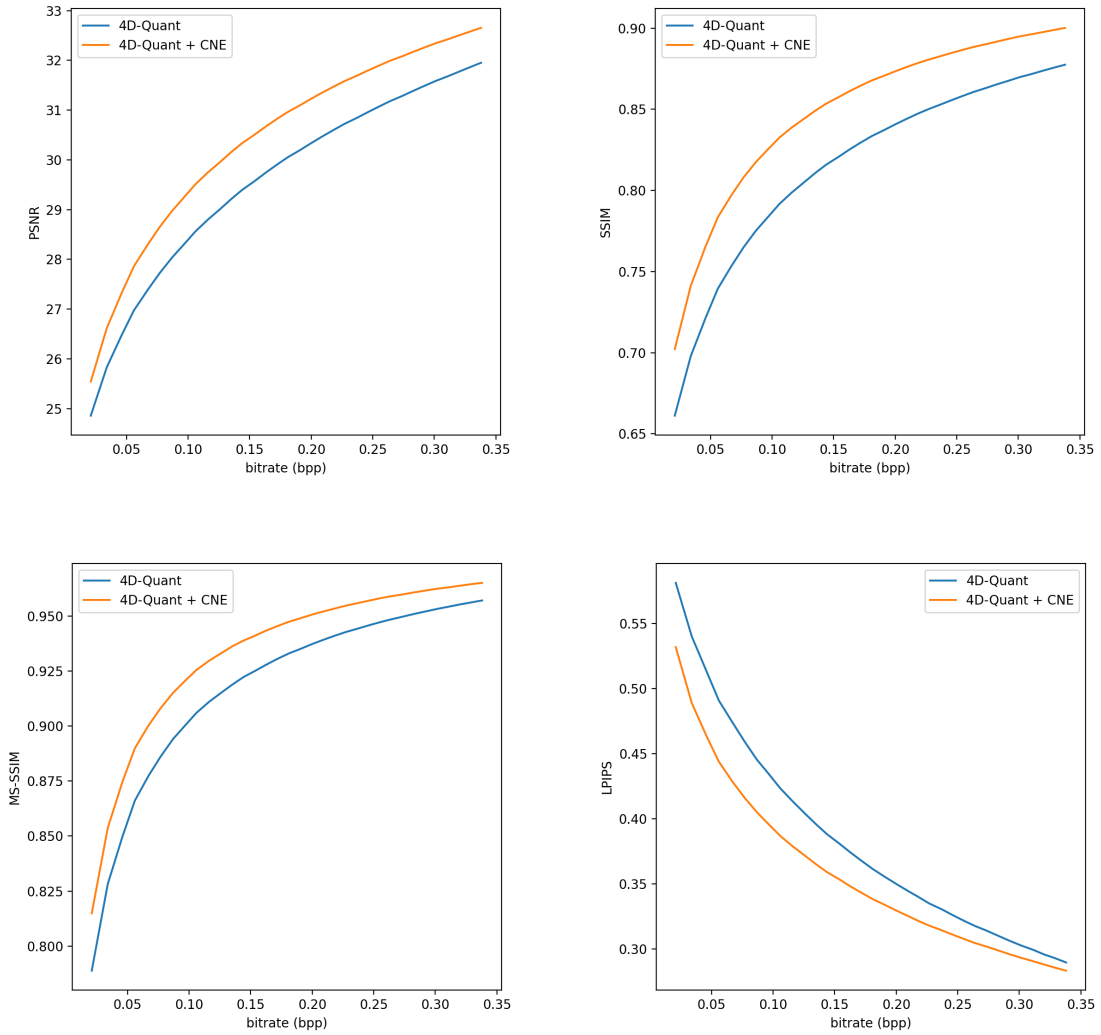


Figure 4.18: Comparison of average PSNR, SSIM, MS-SSIM and LPIPS scores at low bitrate on the LIAM-LF-Dataset.

4.7 Conclusion

Light field datasets will continue to grow as the technology increases in popularity. Until such large datasets exists, we are limited to the use of synthetic samples. We

have generated the largest light field dataset known to date, and have shown it can effectively be used for learning based methods. Furthermore, we have shown evidence that the DR process is successful in the generalization to other domains.

By extending the idea of the JPEG compression standard to higher dimensions, and formalizing it as a neural network, we are able to generate a light field specific quantization table, which we refer to as a quantization tensor. In order to back-propagate through the network, we propose a new approximation to the rounding function given by Equation 4.13, which gives more accurate results at test time than the practices discussed in the current literature.

By incorporating a quality factor parameter to the quantization tensor, we are able to adjust the strength of the compression. For a desired bitrate, we proposed Equation 4.20 for the estimation of the quality factor required.

To further enhance the quality of our quantizer at a given bitrate, we propose our CNE network. The network was trained on aggressively compressed samples from our dataset, using the quantization tensor. By constructing the network to detect features at a variety of resolutions, it is able to enhance pixel-level imperfections such as noise, as well as global artifacts such as blocking. The addition of the CNE improves PSNR scores on average by 0.854 dB, SSIM and MS-SSIM by 0.0338 and 0.0151, and decreases LPIPS by 0.0259 as measured on our dataset across a range of bitrates. Visual comparisons further support the findings, with noticable reduction in common compression artifacts. Similar results were attained when tested on the HCI 4D Dataset.

We conclude that our CNE is a viable tool in the compression of light field data, and outperform modern image compression techniques. It is capable of near perfect

reconstructions at high bitrates, while also producing optimal results at arbitrary small bitrates.

Chapter 5

Conclusion

5.1 Findings

Immersive experiences will likely be the next frontier in digital media, amplifying the need for efficient compression and storage methods. Light fields in particular display large statistical redundancies, due to the similarities of views that can be exploited for substantial reductions in file size.

The combination of ML and CNNs in particular, provide the perfect tool for the task, given a sufficient and diverse enough dataset from which to learn from. As no large light field datasets exist, we explored the use of DR and it's ability to bridge the domain gap; an attempt to create enough variability in the data such that *"the real world may appear to the model as just another variation"* as Tobin et al. [28] describes it.

We tested the idea in the task of image classification. Our study indicated that the technique was capable of transferring well to a variety of different domains.

Furthermore, the results suggested that the primary factor related to classification accuracy was the variety of class subjects. Visual explanation models produced heatmaps which validated the classifier was prioritizing the relevant regions of each images.

Extending the use of DR, we simulated 20,000 light fields with angular resolutions of 9×9. A large range of parameters were randomized, namely lighting, diffusion maps, textures, occluders and camera positions. Using the dataset, we trained a neural network to calculate the entries of a 4-dimensional quantization tensor Q for the associated DCT coefficients, which minimized the reconstructed distortion while inducing sparsity in the encoding. We were able to extrapolate the quantization tensor to higher (and lower) compression ratios by scaling it using a quality factor parameter q. Our quantizer was able to produce higher quality encodings at lower bitrates than the leading image compression algorithms. Unlike JPEG, we were able to formalize a method for the selection of the quality factor parameter given a target bitrate.

We built on our quantizer, by adding a CNE with the goal of reducing compression artifacts and noise in our reconstructed light fields. As we theorized, the convolutional-based model was capable of detecting blocking and ringing artifacts, and successfully degraded them. Similarity and perceptual metrics showed improvements in image quality across the board, on multiple datasets. Visual inspection reinforced these findings. Furthermore, the high performance on the HCI 4D Dataset further provided evidence that the DR technique was an effective strategy when data is scarce.

5.2 Future Work

Equation 4.15 defines the objective function on which our model is optimized on. The λ parameter here governs the tradeoff we assign between distortion and bitrate terms. We finetuned this parameter to a value of 10^{-4}, as it appeared to balance the two terms in the early stages of training and encouraged convergence. Similar to the practice of adjusting the learning rate when stuck in a local minima, it may be beneficial to iteratively increment λ at train time, encouraging similar distortion results at lower bitrates. Furthermore, the quantization tensor Q is sub-optimal for $q \neq 1$, for a given λ. This means that while we can adjust q for more aggressive compression, there exists another quantization tensor Q^* with $q = 1$ that is optimal based on the tradeoff determined by λ. However, as mentioned before, this requires the generation and storage of separate quantization tensors for each bitrate, making it practically infeasible. Nevertheless, further experimentation with λ adjustments are worth exploring.

Another parameter we fixed was the quality factor q of the compressed data, when training our CNE. We made this choice with the intention for our model to generalize to a large range of different compression levels. However, over a consistently narrow range of bitrates, a new CNE can be trained that emphasizes compression artifacts present specifically at those scales. The marginal gains across a set of CNEs trained on other quality factors would be of interest.

Lastly, Equation 4.20 can be further improved, based on the target domain. The relationship between q and b is likely different for natural images, so a more accurate estimate can be determined, once a large enough dataset exist.

Bibliography

[1] Lior Yariv et al. "Volume rendering of neural implicit surfaces". In: *Thirty-Fifth Conference on Neural Information Processing Systems*. 2021.

[2] Amit Raj et al. "ANR: Articulated Neural Rendering for Virtual Avatars". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 3722–3731.

[3] Pratul P. Srinivasan et al. "Learning to Synthesize a 4D RGBD Light Field from a Single Image". In: *International Conference on Computer Vision (ICCV)* (2017).

[4] Edward H. Adelson and James R. Bergen. "The plenoptic function and the elements of early vision". In: *Computational Models of Visual Processing*. MIT Press, 1991, pp. 3–20.

[5] Andreı Gershun. "The light field". In: *Journal of Mathematics and Physics* 18.1-4 (1939), pp. 51–151.

[6] Marc Levoy and Pat Hanrahan. "Light Field Rendering". In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '96. New York, NY, USA: Association for Computing Machinery,

1996, pp. 31–42. ISBN: 0897917464. DOI: 10.1145/237170.237199. URL: https://doi.org/10.1145/237170.237199.

[7] Steven J. Gortler et al. "The Lumigraph". In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '96. New York, NY, USA: Association for Computing Machinery, 1996, pp. 43–54. ISBN: 0897917464. DOI: 10.1145/237170.237200. URL: https://doi.org/10.1145/237170.237200.

[8] P. Moon and D. E. Spencer. *The photic field*. MIT Press, 1981.

[9] Emilio Camahort, Apostolos Lerios, and Donald Fussell. "Uniformly Sampled Light Fields". In: Jan. 1998, pp. 117–130. DOI: 10.1007/978-3-7091-6453-2_11.

[10] Donald G. Dansereau. *Stanford Light Field Archives*. http://lightfields.stanford.edu/. 2018. URL: http://lightfields.stanford.edu/.

[11] G Lippmann. "La photographie intégrale CR Séances Acad". In: *Sci* 146 (1908), pp. 446–51.

[12] E.H. Adelson and J.Y.A. Wang. "Single lens stereo with a plenoptic camera". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 99–106. DOI: 10.1109/34.121783.

[13] Ren Ng et al. "Light Field Photography with a Hand-held Plenoptic Camera". PhD thesis. Stanford University, 2005.

[14] Christian Perwass and Lennart Wietzke. "Single Lens 3D-Camera with Extended Depth-of-Field". In: *Proc. SPIE* 8291 (Feb. 2012), pp. 4–. DOI: 10.1117/12.909882.

[15] Todor Georgiev and Andrew Lumsdaine. "Focused plenoptic camera and rendering". In: *J. Electronic Imaging* 19 (Apr. 2010), p. 021106. DOI: 10.1117/1.3442712.

[16] Bennett Wilburn et al. "High Performance Imaging Using Large Camera Arrays". In: *ACM Trans. Graph.* 24.3 (July 2005), pp. 765–776. ISSN: 0730-0301. DOI: 10.1145/1073204.1073259. URL: https://doi.org/10.1145/1073204.1073259.

[17] Bennett Wilburn et al. "Light field video camera". In: *IS&T/SPIE Electronic Imaging*. 2001.

[18] B. Wilburn et al. "High-speed videography using a dense camera array". In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 2. 2004, pp. II–II. DOI: 10.1109/CVPR.2004.1315176.

[19] Michael Broxton et al. "A Low Cost Multi-Camera Array for Panoramic Light Field Video Capture". In: *SIGGRAPH Asia 2019 Posters*. SA '19. Brisbane, QLD, Australia: Association for Computing Machinery, 2019. ISBN: 9781450369435. DOI: 10.1145/3355056.3364593. URL: https://doi.org/10.1145/3355056.3364593.

[20] David Koller et al. "Protected Interactive 3D Graphics via Remote Rendering". In: *ACM Trans. Graph.* 23.3 (Aug. 2004), pp. 695–703. ISSN: 0730-0301. DOI:

10.1145/1015706.1015782. URL: https://doi.org/10.1145/1015706.1015782.

[21] Ryan S. Overbeck et al. "A System for Acquiring, Processing, and Rendering Panoramic Light Field Stills for Virtual Reality". In: *ACM Trans. Graph.* 37.6 (Dec. 2018). ISSN: 0730-0301. DOI: 10.1145/3272127.3275031. URL: https://doi.org/10.1145/3272127.3275031.

[22] Ben Mildenhall et al. "Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines". In: *ACM Trans. Graph.* 38.4 (July 2019). ISSN: 0730-0301. DOI: 10.1145/3306346.3322980. URL: https://doi.org/10.1145/3306346.3322980.

[23] Chia-Kai Liang and Ravi Ramamoorthi. "A Light Transport Framework for Lenslet Light Field Cameras". In: *ACM Trans. Graph.* 34.2 (Mar. 2015). ISSN: 0730-0301. DOI: 10.1145/2665075. URL: https://doi.org/10.1145/2665075.

[24] Lixin Shi et al. "Light Field Reconstruction Using Sparsity in the Continuous Fourier Domain". In: *ACM Trans. Graph.* 34.1 (Dec. 2015). ISSN: 0730-0301. DOI: 10.1145/2682631. URL: https://doi.org/10.1145/2682631.

[25] Mattia Rossi and Pascal Frossard. "Graph-based light field super-resolution". In: *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*. 2017, pp. 1–6. DOI: 10.1109/MMSP.2017.8122224.

[26] Youngjin Yoon et al. "Learning a Deep Convolutional Network for Light-Field Image Super-Resolution". In: *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*. 2015, pp. 57–65. DOI: 10.1109/ICCVW.2015.17.

[27] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. *Accurate Image Super-Resolution Using Very Deep Convolutional Networks*. 2016. arXiv: 1511.04587 [cs.CV].

[28] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 23–30. DOI: 10.1109/IROS.2017.8202133.

[29] C. E. Shannon. "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.4 (1948), pp. 623–656. DOI: 10.1002/j.1538-7305.1948.tb00917.x.

[30] G.K. Wallace. "The JPEG still picture compression standard". In: *IEEE Transactions on Consumer Electronics* 38.1 (1992), pp. xviii–xxxiv. DOI: 10.1109/30.125072.

[31] Mohammed Ebrahim Al-Mualla, C. Nishan Canagarajah, and David R. Bull. "Chapter 2 - Video Coding: Fundamentals". In: *Video Coding for Mobile Communications*. Ed. by Mohammed Ebrahim Al-Mualla, C. Nishan Canagarajah, and David R. Bull. Signal Processing and its Applications. San Diego: Academic Press, 2002, pp. 9–42. ISBN: 978-0-12-053079-3. DOI: https://doi.org/10.1016/B978-012053079-3/50004-4. URL: https://www.sciencedirect.com/science/article/pii/B9780120530793500044.

[32] K.R. Rao and P. Yip. "CHAPTER 5 - TWO-DIMENSIONAL DCT ALGORITHMS". In: *Discrete Cosine Transform*. Ed. by K.R. RAO and P. YIP. San Diego: Academic Press, 1990, pp. 88–121. ISBN: 978-0-08-092534-9. DOI:

https://doi.org/10.1016/B978-0-08-092534-9.50011-4. URL: https://www.sciencedirect.com/science/article/pii/B9780080925349500114.

[33] Richard F. Haines and Sherry L. Chuang. "The effects of video compression on acceptability of images for monitoring life sciences experiments". In: 1992.

[34] Zoran Kotevski and Pece Mitrevski. "Experimental Comparison of PSNR and SSIM Metrics for Video Quality Estimation". In: *ICT Innovations 2009*. Ed. by Danco Davcev and Jorge Marx Gómez. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 357–366. ISBN: 978-3-642-10781-8.

[35] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[36] Z. Wang, E.P. Simoncelli, and A.C. Bovik. "Multiscale structural similarity for image quality assessment". In: *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*. Vol. 2. 2003, 1398–1402 Vol.2. DOI: 10.1109/ACSSC.2003.1292216.

[37] Richard Zhang et al. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 586–595.

[38] Maxime Wabartha et al. "Handling Black Swan Events in Deep Learning with Diversely Extrapolated Neural Networks". In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. Ed. by Christian Bessiere. Main track. International Joint Conferences on Artificial

Intelligence Organization, July 2020, pp. 2140–2147. DOI: 10.24963/ijcai.2020/296. URL: https://doi.org/10.24963/ijcai.2020/296.

[39] J. Tremblay et al. "Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018, pp. 1082–10828. DOI: 10.1109/CVPRW.2018.00143.

[40] Alireza Shafaei, J.J. Little, and Mark Schmidt. "Play and Learn: Using Video Games to Train Computer Vision Models". In: Jan. 2016, pp. 26.1–26.13. DOI: 10.5244/C.30.26.

[41] A. Atapour-Abarghouei and T. P. Breckon. "Real-Time Monocular Depth Estimation Using Synthetic Data with Domain Adaptation via Image Style Transfer". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2800–2810. DOI: 10.1109/CVPR.2018.00296.

[42] Jacob Shermeyer et al. *RarePlanes: Synthetic Data Takes Flight*. 2020. arXiv: 2006.02963 [cs.CV].

[43] X. Yue et al. "Domain Randomization and Pyramid Consistency: Simulation-to-Real Generalization Without Accessing Target Domain Data". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 2100–2110. DOI: 10.1109/ICCV.2019.00219.

[44] Yair Movshovitz-Attias, Takeo Kanade, and Yaser Sheikh. "How Useful Is Photo-Realistic Rendering for Visual Learning?" In: *Computer Vision – ECCV 2016 Workshops*. Ed. by Gang Hua and Hervé Jégou. Cham: Springer International Publishing, 2016, pp. 202–217. ISBN: 978-3-319-49409-8.

[45] J. Tobin et al. "Domain Randomization and Generative Models for Robotic Grasping". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 3482–3489. DOI: 10.1109/IROS.2018.8593933.

[46] Antonio Loquercio et al. "Deep Drone Racing: From Simulation to Reality With Domain Randomization". In: *IEEE Transactions on Robotics* 36.1 (Feb. 2020), pp. 1–14. ISSN: 1941-0468. DOI: 10.1109/tro.2019.2942989. URL: http://dx.doi.org/10.1109/TRO.2019.2942989.

[47] Stefan Hinterstoisser et al. "On Pre-trained Image Features and Synthetic Images for Deep Learning". In: *Computer Vision – ECCV 2018 Workshops*. Ed. by Laura Leal-Taixé and Stefan Roth. Cham: Springer International Publishing, 2019, pp. 682–697. ISBN: 978-3-030-11009-3.

[48] Xingchao Peng et al. "Exploring Invariances in Deep Convolutional Neural Networks Using Synthetic Images". In: *CoRR* abs/1412.7122 (2014). arXiv: 1412.7122. URL: http://arxiv.org/abs/1412.7122.

[49] W. Chen et al. "Synthesizing Training Images for Boosting Human 3D Pose Estimation". In: *2016 Fourth International Conference on 3D Vision (3DV)*. 2016, pp. 479–488. DOI: 10.1109/3DV.2016.58.

[50] Bojan Pepik et al. "What Is Holding Back Convnets for Detection?" In: *Pattern Recognition*. Ed. by Juergen Gall, Peter Gehler, and Bastian Leibe. Cham: Springer International Publishing, 2015, pp. 517–528. ISBN: 978-3-319-24947-6.

[51] X. Peng et al. "Learning Deep Object Detectors from 3D Models". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1278–1286. DOI: 10.1109/ICCV.2015.151.

[52] M. Mozian et al. "Learning Domain Randomization Distributions for Training Robust Locomotion Policies". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 6112–6117. DOI: 10.1109/IROS45743.2020.9341019.

[53] Bhairav Mehta et al. *Active Domain Randomization*. 2019. arXiv: 1904.04762 [cs.LG].

[54] Aayush Prakash et al. "Structured Domain Randomization: Bridging the Reality Gap by Context-Aware Synthetic Data". In: May 2019, pp. 7249–7255. DOI: 10.1109/ICRA.2019.8794443.

[55] OpenAI et al. *Solving Rubik's Cube with a Robot Hand*. 2019. arXiv: 1910.07113 [cs.LG].

[56] Dominik Schraml. "Physically based synthetic image generation for machine learning: a review of pertinent literature". In: *Photonics and Education in Measurement Science 2019*. Ed. by Maik Rosenberger, Paul-Gerald Dittrich, and Bernhard Zagar. Vol. 11144. International Society for Optics and Photonics. SPIE, 2019, pp. 108–120. DOI: 10.1117/12.2533485. URL: https://doi.org/10.1117/12.2533485.

[57] Kaggle. *Dogs vs Cats Dataset*. 2013. URL: https://www.kaggle.com/c/dogs-vs-cats/data.

[58] R. R. Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74.

[59] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 818–833. ISBN: 978-3-319-10590-1.

[60] Andreas Opelt and Axel Pinz. *GRAZ-02 Dataset*. 2004. URL: https://www-old.emt.tugraz.at/~pinz/data/GRAZ_02/.

[61] Shahid Khan. *Bicycle Image Dataset*. June 2020. URL: http://maviintelligence.com/bicycle-image-dataset/.

[62] Jonathan Krause et al. "3D Object Representations for Fine-Grained Categorization". In: *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia, 2013.

[63] Luis da Silva Cruz et al. "JPEG Pleno: Standardizing a Coding Framework and Tools for Plenoptic Imaging Modalities". In: *ITU Journal: ICT Discoveries, Special issue on The Future of Video and Immersive Media* 3 (June 2020).

[64] Dong Liu et al. "Pseudo-sequence-based light field image compression". In: *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*. 2016, pp. 1–4. DOI: 10.1109/ICMEW.2016.7574674.

[65] Yun Li, Roger Olsson, and Mårten Sjöström. "Compression of unfocused plenoptic images using a displacement intra prediction". In: *2016 IEEE International*

Conference on Multimedia Expo Workshops (ICMEW). 2016, pp. 1–4. DOI: 10.1109/ICMEW.2016.7574673.

[66] Ricardo J. S. Monteiro et al. "Light field HEVC-based image coding using locally linear embedding and self-similarity compensated prediction". In: *2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)* (2016), pp. 1–4.

[67] Caroline Conti, Paulo Nunes, and Luís Ducla Soares. "HEVC-based light field image coding with bi-predicted self-similarity compensation". In: *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*. 2016, pp. 1–4. DOI: 10.1109/ICMEW.2016.7574667.

[68] Xiaoran Jiang, Mikaël Le Pendu, and Christine Guillemot. "Light field compression using depth image based view synthesis". In: *2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*. 2017, pp. 19–24. DOI: 10.1109/ICMEW.2017.8026313.

[69] João M. Santos et al. "Lossless light-field compression using reversible colour transformations". In: *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*. 2017, pp. 1–6. DOI: 10.1109/IPTA.2017.8310154.

[70] João M. Santos et al. "Lossless Compression of Light Fields Using Multi-reference Minimum Rate Predictors". In: *2019 Data Compression Conference (DCC)*. 2019, pp. 408–417. DOI: 10.1109/DCC.2019.00049.

[71] Milan Stepanov et al. "Learning-based lossless light field compression". In: *IEEE International Workshop on Multimedia Signal Processing (MMSP'2021)*.

Tampere, Finland, Oct. 2021. URL: https://hal.archives-ouvertes.fr/hal-03298954.

[72] J. Chen, J. Hou, and L. Chau. "Light Field Compression With Disparity-Guided Sparse Coding Based on Structural Key Views". In: *IEEE Transactions on Image Processing* 27.1 (Jan. 2018), pp. 314–324. ISSN: 1057-7149. DOI: 10.1109/TIP.2017.2750413.

[73] Ioan Tabus, Petri Helin, and Pekka Astola. "Lossy compression of lenslet images from plenoptic cameras combining sparse predictive coding and JPEG 2000". In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, pp. 4567–4571. DOI: 10.1109/ICIP.2017.8297147.

[74] X. Zhang et al. "Surface Light Field Compression Using a Point Cloud Codec". In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9.1 (Mar. 2019), pp. 163–176. ISSN: 2156-3357. DOI: 10.1109/JETCAS.2018.2883479.

[75] Kshitij Marwah et al. "Compressive Light Field Photography Using Overcomplete Dictionaries and Optimized Projections". In: 32.4 (July 2013). ISSN: 0730-0301. DOI: 10.1145/2461912.2461914. URL: https://doi.org/10.1145/2461912.2461914.

[76] Ehsan Miandji, Saghi Hajisharif, and Jonas Unger. "A Unified Framework for Compression and Compressed Sensing of Light Fields and Light Field Videos". In: *ACM Trans. Graph.* 38.3 (May 2019). ISSN: 0730-0301. DOI: 10.1145/3269980. URL: https://doi.org/10.1145/3269980.

[77] Xiaoran Jiang et al. "Light Field Compression With Homography-Based Low-Rank Approximation". In: *IEEE Journal of Selected Topics in Signal Processing* 11.7 (2017), pp. 1132–1145. DOI: 10.1109/JSTSP.2017.2747078.

[78] Zhenghui Zhao et al. "Light Field Image Compression Based on Deep Learning". In: *2018 IEEE International Conference on Multimedia and Expo (ICME)*. 2018, pp. 1–6. DOI: 10.1109/ICME.2018.8486546.

[79] Jinbo Zhao et al. "Light Field Image Sparse Coding via CNN-Based EPI Super-Resolution". In: *2018 IEEE Visual Communications and Image Processing (VCIP)*. 2018, pp. 1–4. DOI: 10.1109/VCIP.2018.8698714.

[80] N. Bakir et al. "Light Field Image Compression Based on Convolutional Neural Networks and Linear Approximation". In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. Oct. 2018, pp. 1128–1132. DOI: 10.1109/ICIP.2018.8451597.

[81] M. Gupta et al. "Compressive Light Field Reconstructions Using Deep Learning". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. July 2017, pp. 1277–1286. DOI: 10.1109/CVPRW.2017.168.

[82] Mohana Singh and Renu M. Rameshan. *Learning-Based Practical Light Field Image Compression Using A Disparity-Aware Model*. 2021. arXiv: 2106.11558 [eess.IV].

[83] David Minnen and Saurabh Singh. "Channel-Wise Autoregressive Entropy Models for Learned Image Compression". In: *2020 IEEE International Conference on Image Processing (ICIP)*. 2020, pp. 3339–3343. DOI: 10.1109/ICIP40778.2020.9190935.

[84] Chuanmin Jia et al. "Light Field Image Compression Using Generative Adversarial Network-Based View Synthesis". In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9.1 (2019), pp. 177–189. DOI: 10.1109/JETCAS.2018.2886642.

[85] Deyang Liu et al. "View Synthesis-based Light Field Image Compression Using a Generative Adversarial Network". In: *Information Sciences* 545 (Aug. 2020). DOI: 10.1016/j.ins.2020.07.073.

[86] Sharon Zhou et al. "HYPE: A Benchmark for Human Eye Perceptual Evaluation of Generative Models". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[87] Jin Wang et al. "Light Field Image Compression Using Multi-branch Spatial Transformer Networks Based View Synthesis". In: *2020 Data Compression Conference (DCC)*. 2020, pp. 397–397. DOI: 10.1109/DCC47342.2020.00047.

[88] Ben Mildenhall et al. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *ECCV*. 2020.

[89] Matthew Tancik et al. "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains". In: *NeurIPS* (2020).

[90] Alex Yu and Sara Fridovich-Keil et al. *Plenoxels: Radiance Fields without Neural Networks*. 2021. arXiv: 2112.05131 [cs.CV].

[91] Stephan J. Garbin et al. "FastNeRF: High-Fidelity Neural Rendering at 200FPS". In: (Oct. 2021), pp. 14346–14355.

[92] Mojtaba Bemana et al. "X-Fields: Implicit Neural View-, Light- and Time-Image Interpolation". In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2020)* 39.6 (2020). DOI: 10.1145/3414685.3417827.

[93] Jonathan T. Barron et al. "Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields". In: *ICCV* (2021).

[94] Keunhong Park et al. "Nerfies: Deformable Neural Radiance Fields". In: *ICCV* (2021).

[95] Edgar Tretschk et al. "Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene From Monocular Video". In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2021.

[96] Ricardo Martin-Brualla et al. "NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections". In: *CVPR*. 2021.

[97] Hanzhi Fan et al. "Two-stage convolutional neural network for light field super-resolution". In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, pp. 1167–1171. DOI: 10.1109/ICIP.2017.8296465.

[98] Yan Yuan, Ziqi Cao, and Lijuan Su. "Light-Field Image Superresolution Using a Combined Deep CNN Based on EPI". In: *IEEE Signal Processing Letters* 25.9 (2018), pp. 1359–1363. DOI: 10.1109/LSP.2018.2856619.

[99] Yunlong Wang et al. "LFNet: A Novel Bidirectional Recurrent Convolutional Neural Network for Light-Field Image Super-Resolution". In: *IEEE Transactions on Image Processing* 27.9 (2018), pp. 4274–4286. DOI: 10.1109/TIP.2018.2834819.

[100] M. Shahzeb Khan Gul and Bahadir K. Gunturk. "Spatial and Angular Resolution Enhancement of Light Fields Using Convolutional Neural Networks". In: *IEEE Transactions on Image Processing* 27.5 (2018), pp. 2146–2159. DOI: 10.1109/TIP.2018.2794181.

[101] David Alexandre et al. "An Autoencoder-based Learned Image Compressor: Description of Challenge Proposal by NCTU". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2018.

[102] L. Theis et al. "Lossy Image Compression with Compressive Autoencoders". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/pdf?id=rJiNwv9gg.

[103] Mu Li et al. "Learning Convolutional Networks for Content-Weighted Image Compression". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3214–3223. DOI: 10.1109/CVPR.2018.00339.

[104] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. "End-to-end Optimized Image Compression". In: *CoRR* abs/1611.01704 (2016). arXiv: 1611.01704. URL: http://arxiv.org/abs/1611.01704.

[105] Bee Lim et al. "Enhanced Deep Residual Networks for Single Image Super-Resolution". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2017, pp. 1132–1140. DOI: 10.1109/CVPRW.2017.151.

[106] Chao Dong et al. "Image Super-Resolution Using Deep Convolutional Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.2 (2016), pp. 295–307. DOI: 10.1109/TPAMI.2015.2439281.

[107] Chao Dong, Chen Change Loy, and Xiaoou Tang. "Accelerating the Super-Resolution Convolutional Neural Network". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 391–407. ISBN: 978-3-319-46475-6.

[108] Chao Dong et al. "Compression Artifacts Reduction by a Deep Convolutional Network". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 576–584. DOI: 10.1109/ICCV.2015.73.

[109] Lukas Cavigelli, Pascal Alexander Hager, and Luca Benini. "CAS-CNN: A deep convolutional neural network for image compression artifact suppression". In: *2017 International Joint Conference on Neural Networks (IJCNN)* (2017), pp. 752–759.

[110] Christian Ledig et al. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 105–114. DOI: 10.1109/CVPR.2017.19.

[111] A. Mousnier, E. Vural, and C. Guillemot. *Lytro first generation dataset*. 2019. URL: https://www.irisa.fr/temics/demos/lightField/index.html (visited on 05/20/2019).

[112] MIT. *Synthetic Light Field Archive*. 2019. URL: http://web.media.mit.edu/~gordonw/SyntheticLightFields/index.php (visited on 05/20/2019).

[113] Katrin Honauer et al. "A dataset and evaluation methodology for depth estimation on 4D light fields". In: *Asian Conference on Computer Vision*. Springer. 2016.

[114] Nianyi Li et al. "Saliency Detection on Light Field". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2014.

[115] Alireza Ghasemi, Nelly Afonso, and Martin Vetterli. "LCAV-31: A dataset for light field object recognition". In: vol. 9020. Mar. 2013, p. 902014. DOI: 10.1117/12.2041097.

[116] C. Hazirbas et al. "Deep Depth From Focus". In: *Asian Conference on Computer Vision (ACCV)*. Dec. 2018. eprint: 1704.01085. URL: https://hazirbas.com/projects/ddff/.

[117] S. Heber and T. Pock. "Convolutional Networks for Shape from Light Field". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 3746–3754. DOI: 10.1109/CVPR.2016.407.

[118] Vamsi Kiran Adhikarla et al. "Towards a Quality Metric for Dense Light Fields". In: *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[119] Pierre Matysiak et al. "A pipeline for lenslet light field quality enhancement". In: *IEEE International Conference on Image Processing (ICIP 2018)*. Oct. 7, 2018. URL: https://v-sense.scss.tcd.ie:443/wp-content/uploads/2018/05/LFPipeline_ICIP18.pdf. published.

[120] Svetozar Zarko Valtchev and Jianhong Wu. "Domain randomization for neural network classification". In: *Journal of big Data* 8.1 (2021), pp. 1–12.

[121] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001, pp. 71–73.

[122] Leon Lasdon, Richard Fox, and Margery Ratner. "Nonlinear Optimization Using the Generalized Reduced Gradient Method". In: *Revue Française d'Automatique, Informatique, Recherche Opérationnelle. Série Verte* 8 (Oct. 1973), p. 63. DOI: 10.1051/ro/197408V300731.

Appendices

Appendix A

Rights and Permissions

A.1 Springer Journal of Big Data

Springer Nature grants authors the right to use in part or in whole, their own work in their thesis. For more information on author permission, see `https://www.springer.com/gp/rights-permissions/obtaining-permissions/882`

Appendix B

Dataset

B.1 LIAM Light Field Dataset

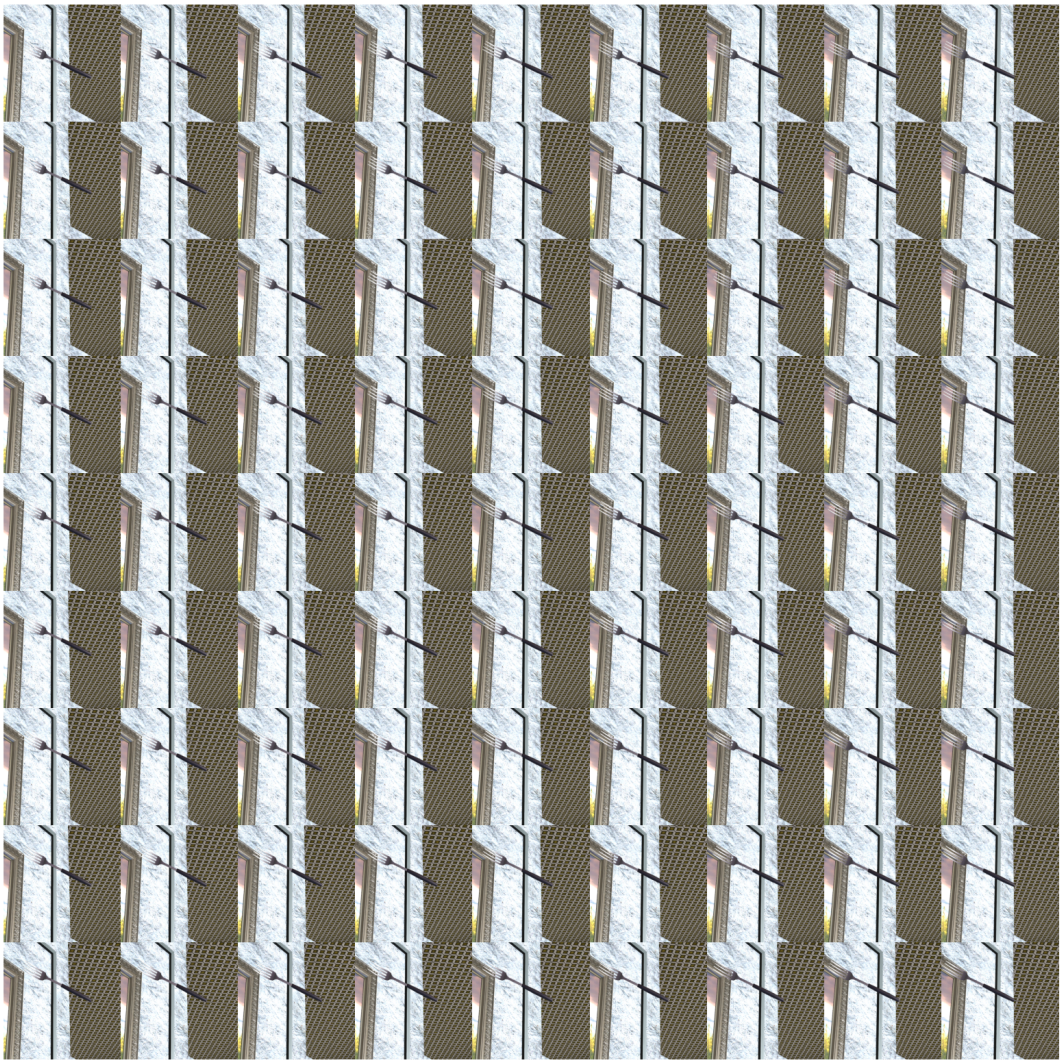


Figure B.1: Example 1. A *uv* array of *st* images.

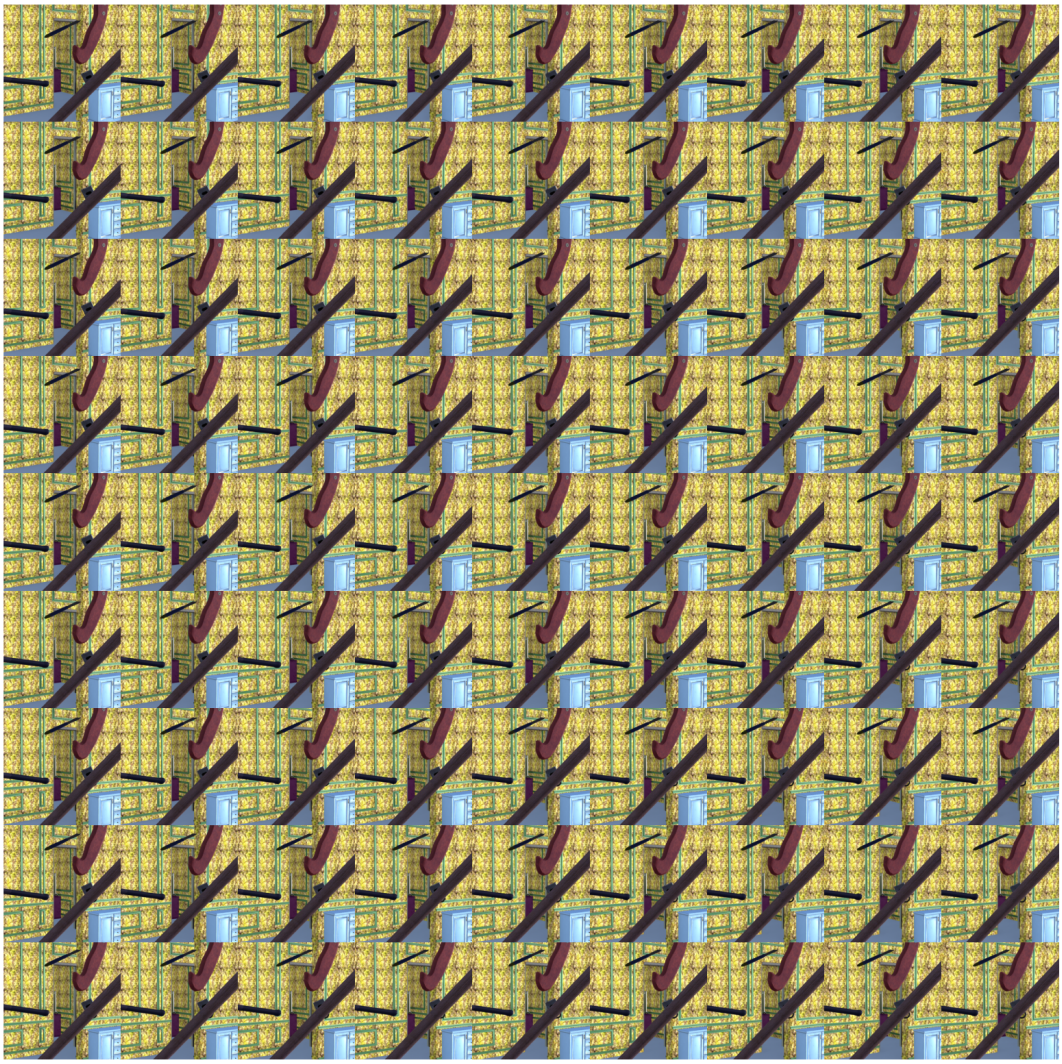


Figure B.2: Example 2. A *uv* array of *st* images.

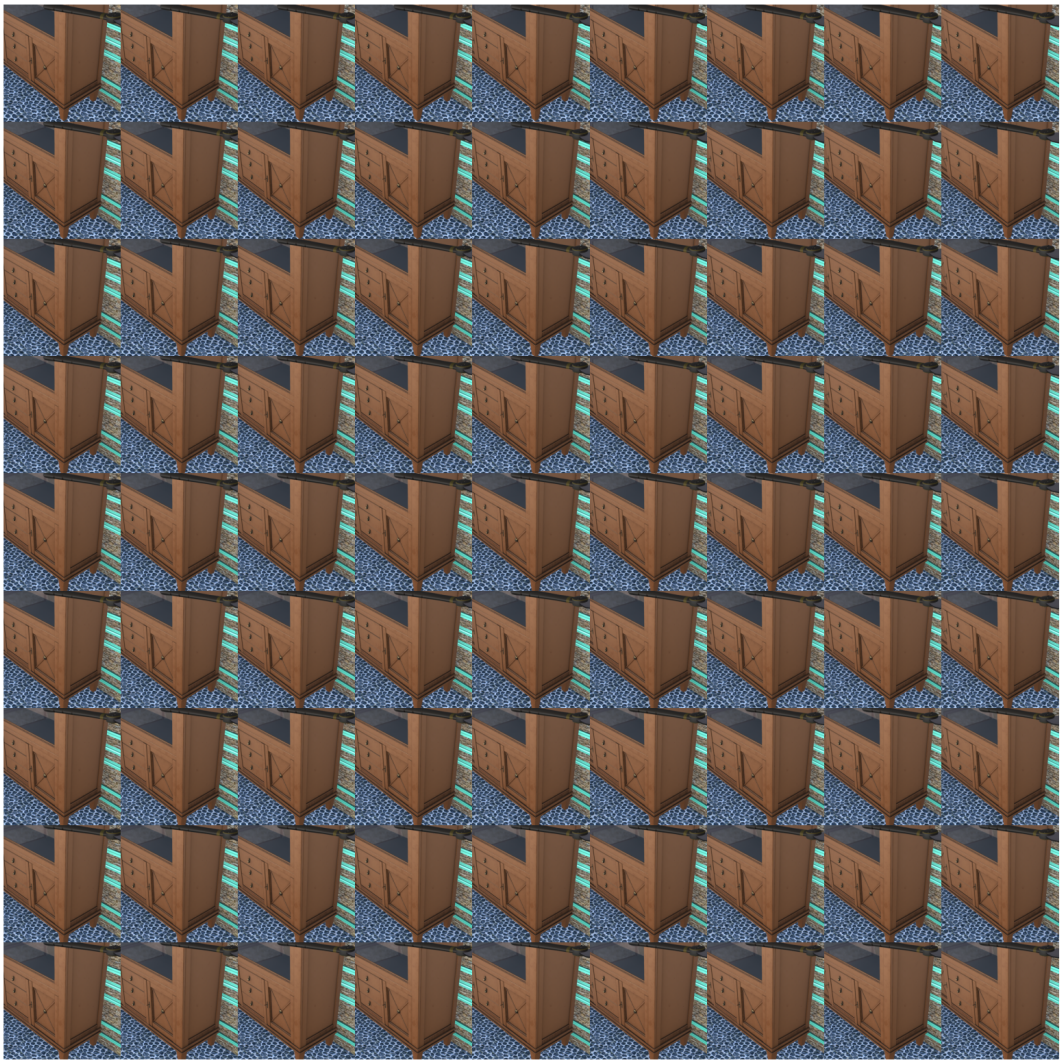


Figure B.3: Example 3. A *uv* array of *st* images.

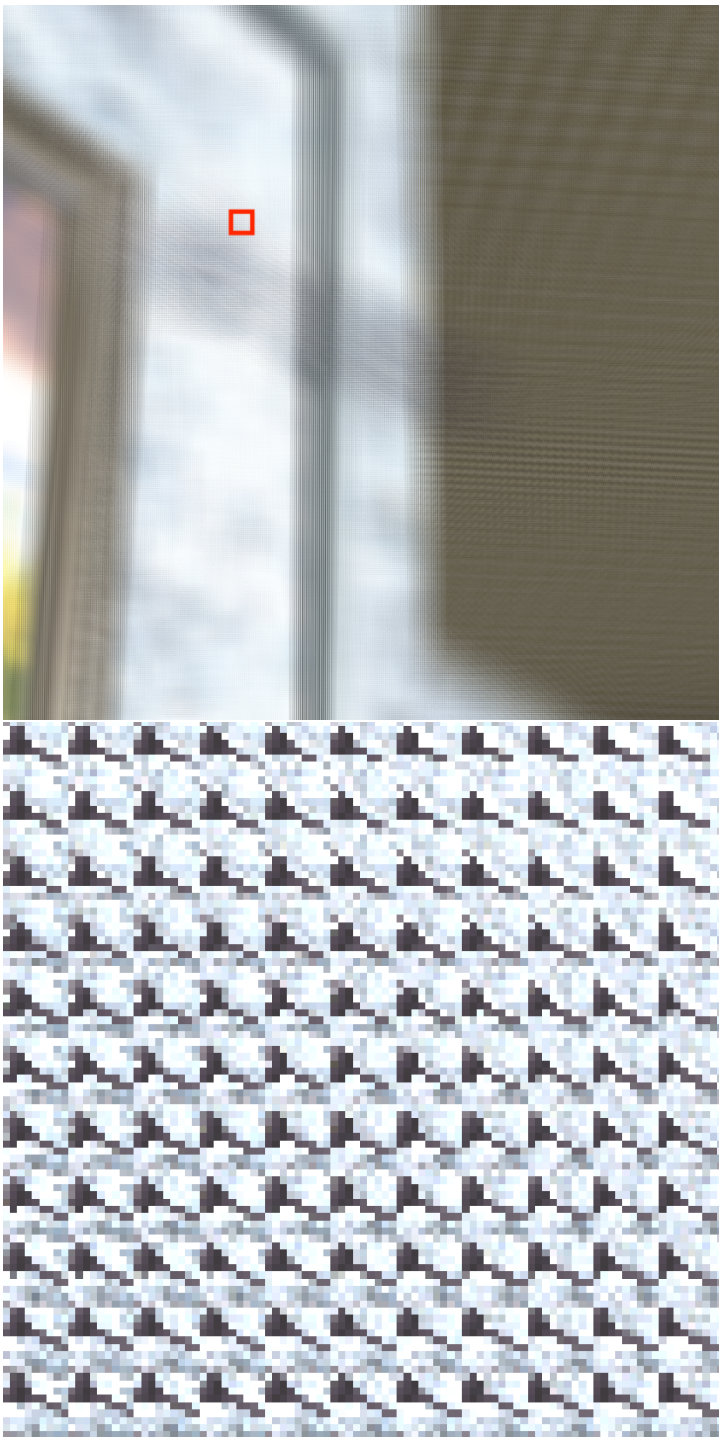


Figure B.4: Example 4. A *st* array of *uv* images. An enlarged view is shown for the region outlined in red on the bottom.

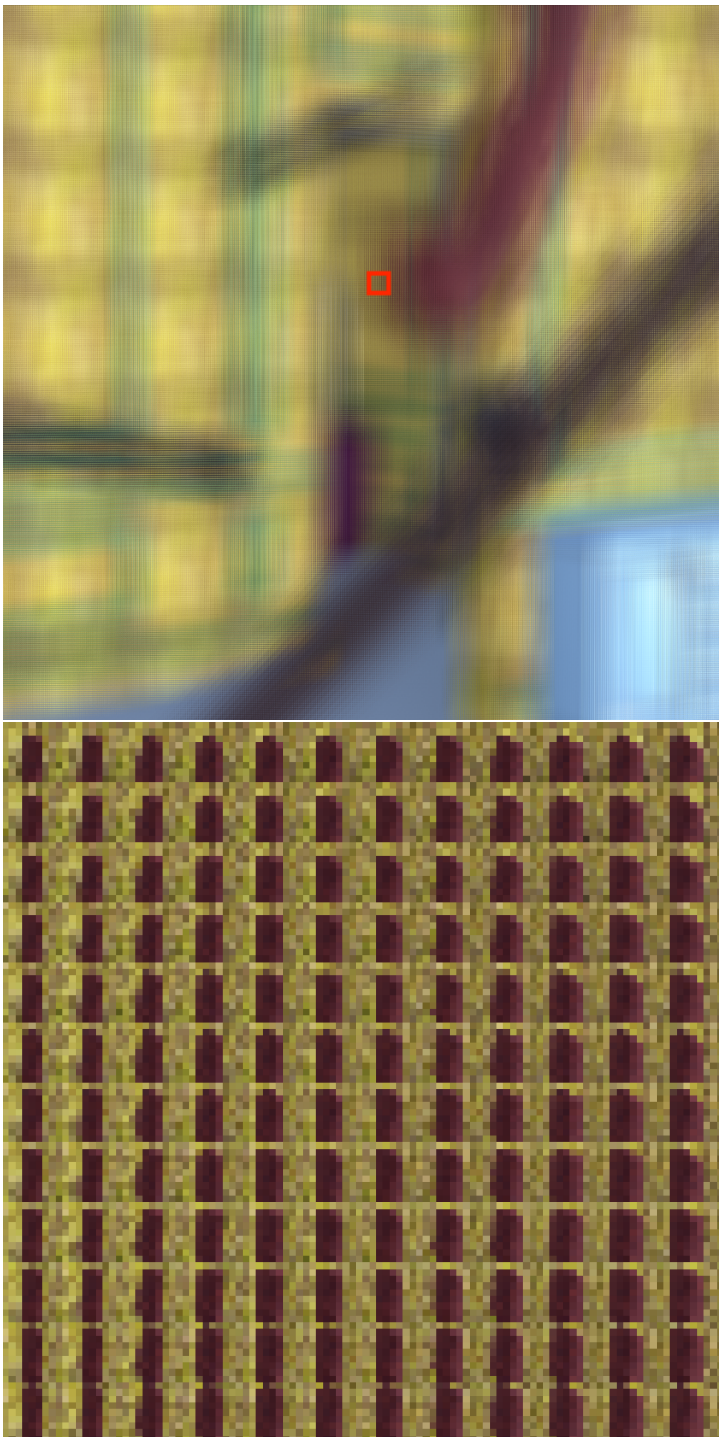


Figure B.5: Example 5. A *st* array of *uv* images. An enlarged view is shown for the region outlined in red on the bottom.

Figure B.6: Example 6. A *st* array of *uv* images. An enlarged view is shown for the region outlined in red on the bottom.

Appendix C

Derivations

C.1 Quantization Approximation Function

The closed form of the series

$$s^*(x) = \frac{1}{\pi} \sum_{n=1}^{\infty} \frac{\tau^{n+1}}{n} \sin(2n\pi x)$$

can be attained as follows:

$$
\begin{aligned}
s^*(x) &= \frac{1}{\pi}\sum_{n=1}^{\infty}\frac{(\tau)^{n+1}}{n}\sin(2n\pi x) \\
&= \frac{1}{\pi}\sum_{n=1}^{\infty}\frac{\tau^n\tau}{n}\sin(2n\pi x) \\
&= \frac{\tau}{\pi}\sum_{n=1}^{\infty}\frac{\tau^n}{n}\sin(2n\pi x) \\
&= \frac{\tau}{n}\sum_{n=1}^{\infty}\frac{\tau^n\,\mathrm{Im}(e^{2\pi nix})}{n} \\
&= \frac{\tau}{n}\,\mathrm{Im}\left(\sum_{n=1}^{\infty}\frac{\tau^n e^{2\pi nix}}{n}\right) \\
&= \frac{\tau}{n}\,\mathrm{Im}\left(\sum_{n=1}^{\infty}\frac{\left(\tau e^{2\pi ix}\right)^n}{n}\right) \\
&= -\frac{\tau}{n}\,\mathrm{Im}\left(\log(1-\tau e^{2\pi ix})\right) \text{ by Taylor series of log} \\
&= -\frac{\tau}{n}\,\mathrm{Im}\left(\log(1-\tau\cos(2\pi x)-i\tau\sin(2\pi x))\right) \\
&= \frac{\tau}{n}\tan^{-1}\left(\frac{\tau\sin(2\pi x)}{1-\tau\cos(2\pi x)}\right)
\end{aligned}
$$

since

$$
\begin{aligned}
\mathrm{Im}\left(\log(x+iy)\right) &= \mathrm{Im}\left(\log(re^{i\theta})\right) \\
&= \mathrm{Im}\left(\log(r)+\log(e^{i\theta})\right) \\
&= \mathrm{Im}\left(\log(r)+i\theta\right) \\
&= \theta \\
&= \tan^{-1}\left(\frac{y}{x}\right)
\end{aligned}
$$

C.2 Derivative of Rounding Function

The derivative of

$$g(x) = x - \frac{\tau}{\pi} \tan^{-1}\left(\frac{\tau \sin(2\pi x)}{1 - \tau\cos(2\pi x)}\right)$$

is

$$\begin{aligned}
g'(x) &= \frac{d}{dx}\left(x - \frac{\tau}{\pi}\tan^{-1}\left(\frac{\tau\sin(2\pi x)}{1-\tau\cos(2\pi x)}\right)\right) \\
&= \frac{d}{dx}(x) - \frac{d}{dx}\left(\frac{\tau}{\pi}\tan^{-1}\left(\frac{\tau\sin(2\pi x)}{1-\tau\cos(2\pi x)}\right)\right) \\
&= 1 - \frac{\tau}{\pi}\left(\frac{1}{\left(\frac{\tau\sin(2\pi x)}{1-\tau\cos(2\pi x)}\right)^2 + 1} \cdot \frac{d}{dx}\left(\frac{\tau\sin(2\pi x)}{1-\tau\cos(2\pi x)}\right)\right) \\
&= 1 - \frac{\tau^2}{\pi}\left(\frac{1}{\left(\frac{\tau\sin(2\pi x)}{1-\tau\cos(2\pi x)}\right)^2 + 1}\right) \cdot \frac{d}{dx}\left(\frac{\sin(2\pi x)}{1-\tau\cos(2\pi x)}\right) \\
&= 1 - \frac{\tau^2}{\pi}\left(\frac{1}{\left(\frac{\tau\sin(2\pi x)}{1-\tau\cos(2\pi x)}\right)^2 + 1}\right) \\
&\qquad \cdot \left(\frac{2\pi\cos(2\pi x)(1-\tau\cos(2\pi x)) - (2\pi\tau\sin^2(2\pi x))}{(1-\tau\cos(2\pi x))^2}\right) \\
&= 1 - \frac{2\tau^2(\cos(2\pi x)(1-\tau\cos(2\pi x)) - \tau\sin^2(2\pi x))}{\tau^2\sin^2(2\pi x) + (1-\tau\cos(2\pi x))^2}
\end{aligned}$$

Appendix D

Compressed Examples

D.1 Examples of 4D Quantizer and Neural Enhancer

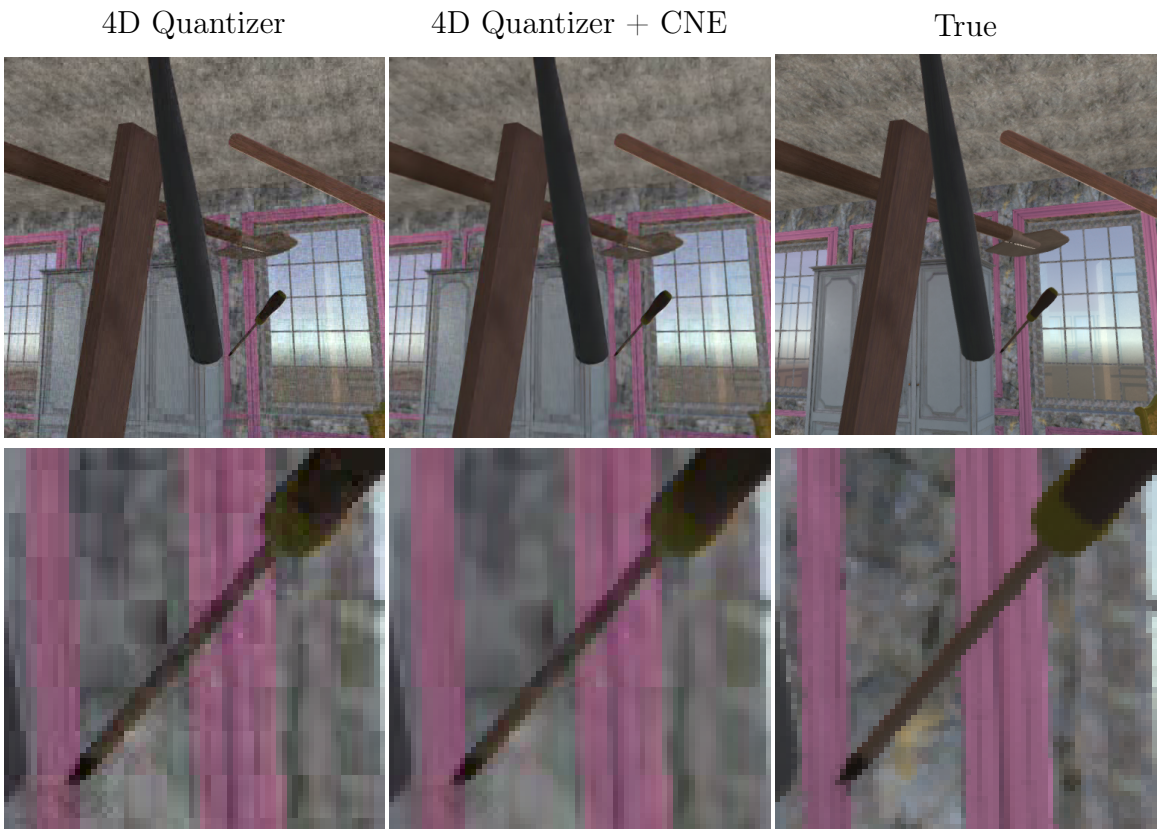


Figure D.1: Example of a scene compressed with our models at a typical bitrate.

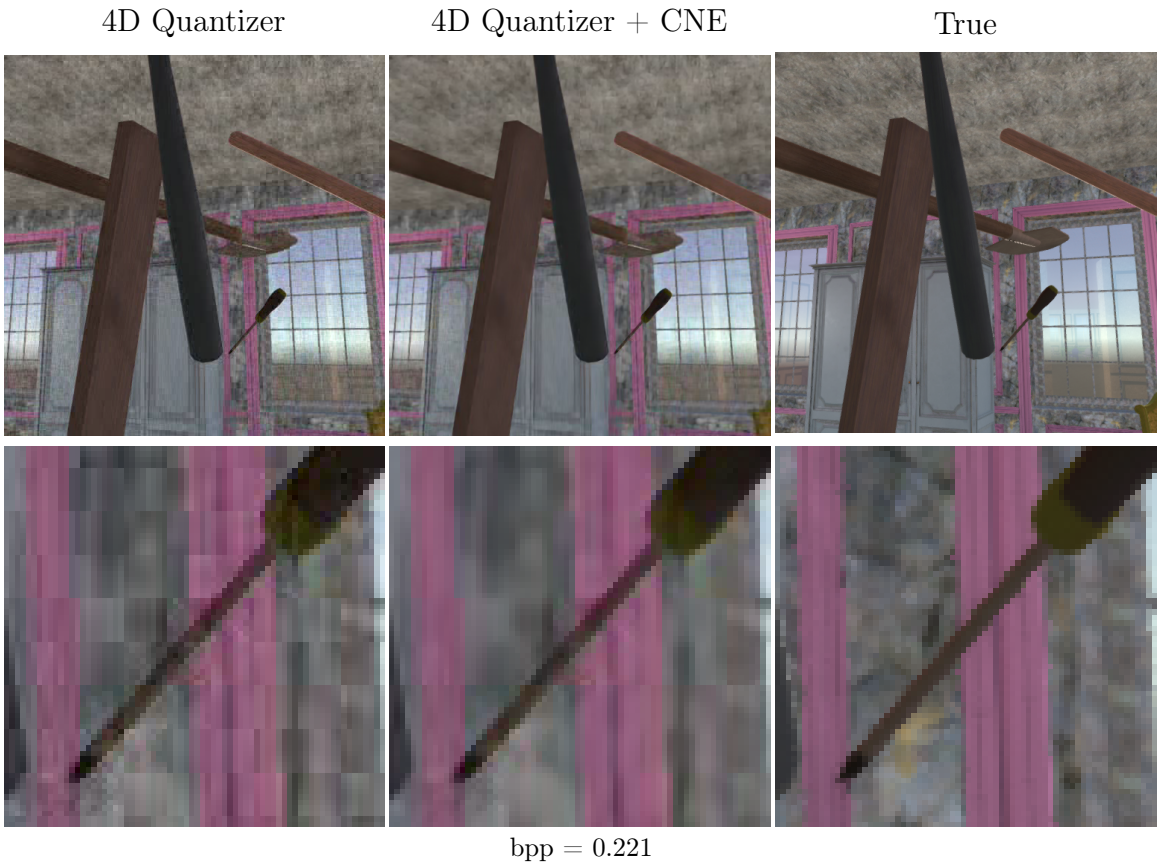


Figure D.2: Example of a scene compressed with our models at a low bitrate.

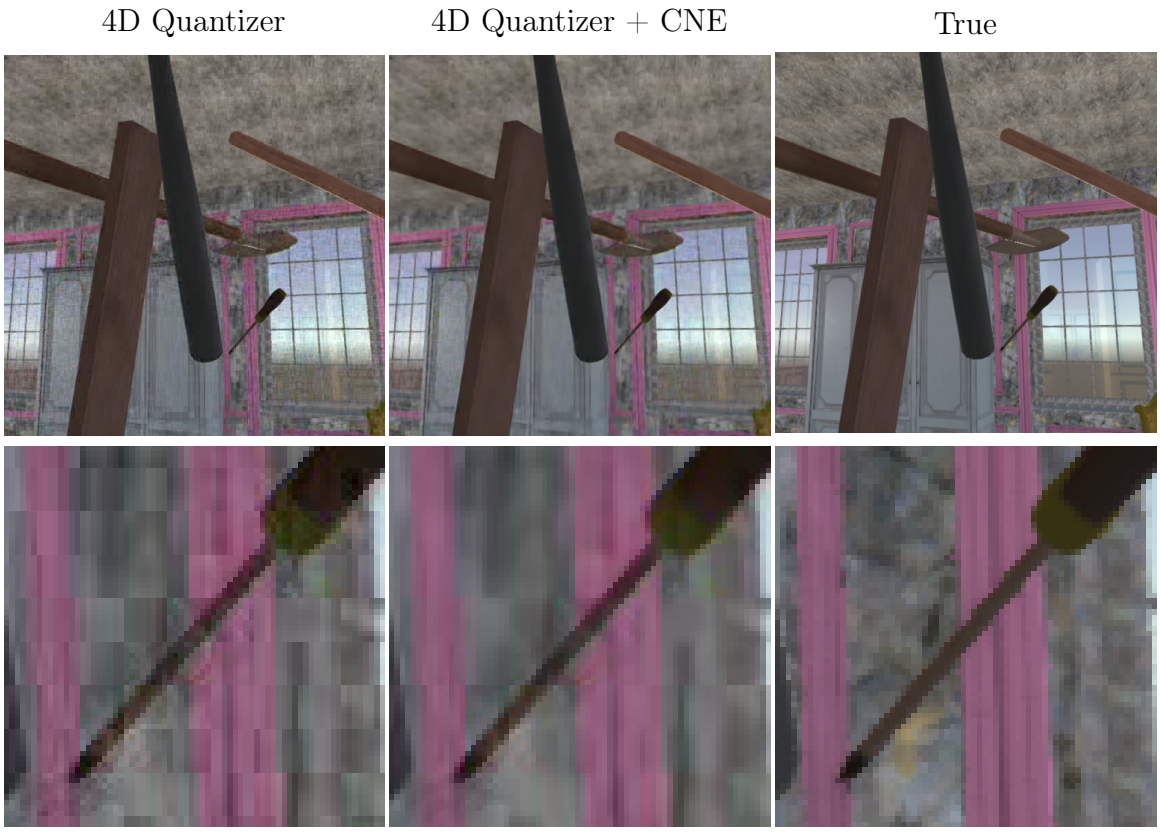


Figure D.3: Example of a scene compressed with our models at a very low bitrate.

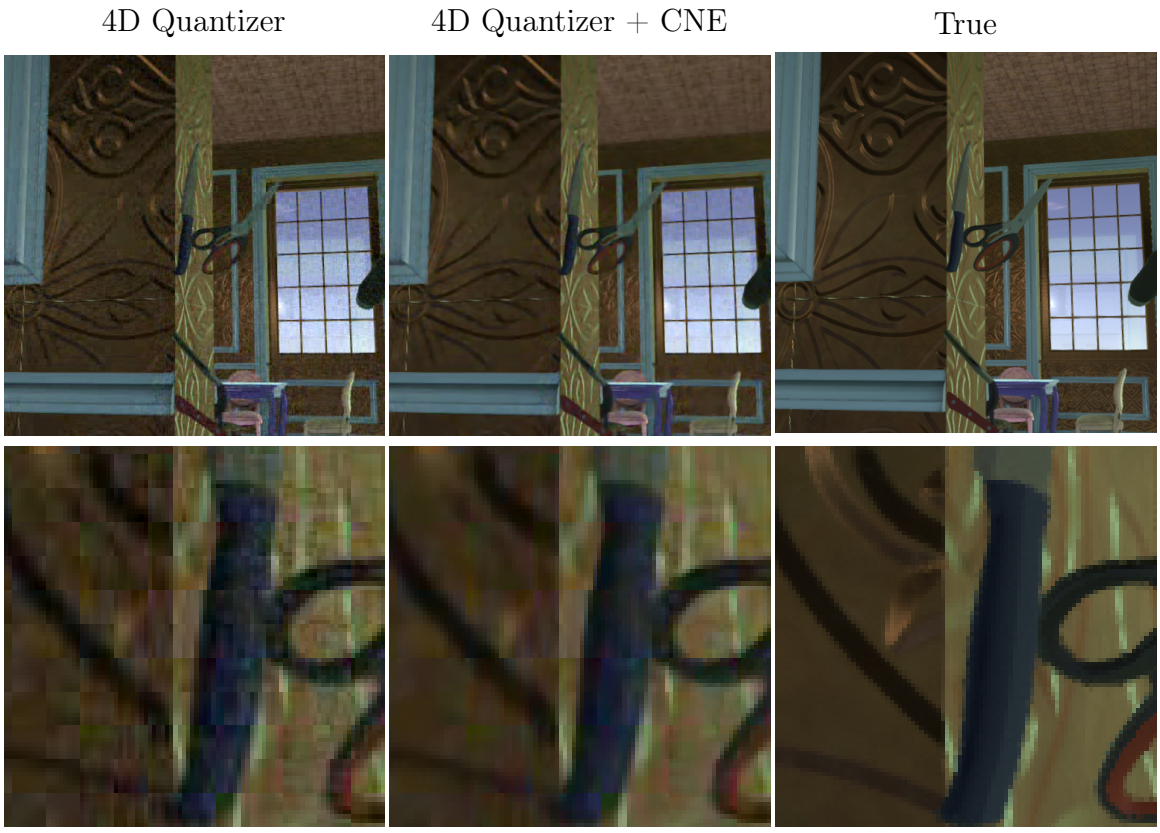


Figure D.4: Example of a scene compressed with our models at a typical bitrate.

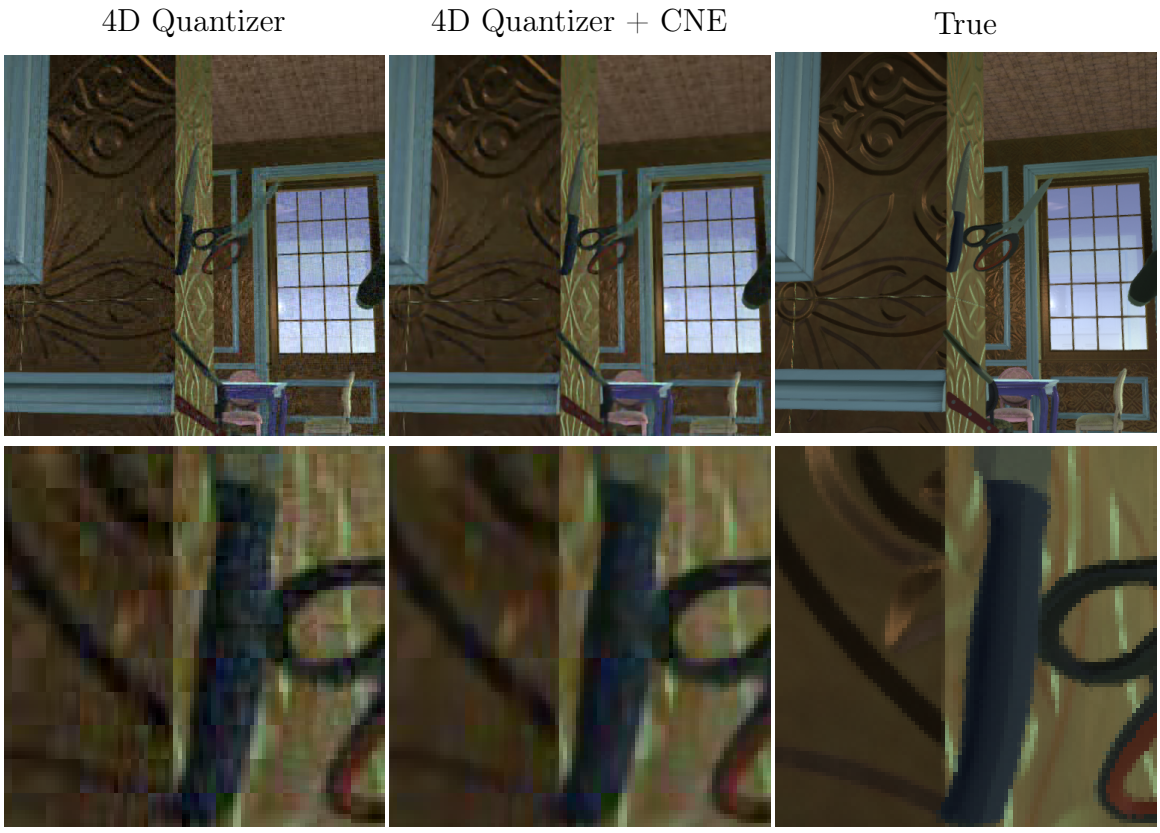


Figure D.5: Example of a scene compressed with our models at a low bitrate.

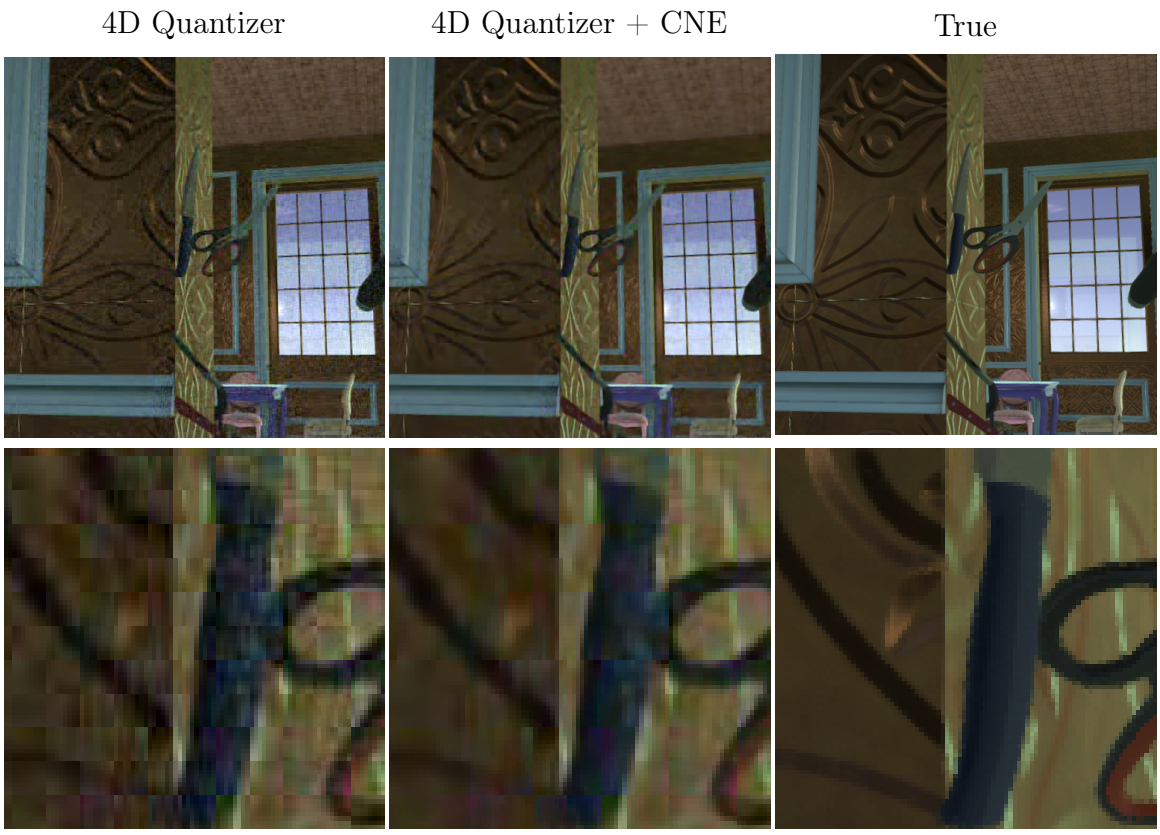


Figure D.6: Example of a scene compressed with our models at a very low bitrate.

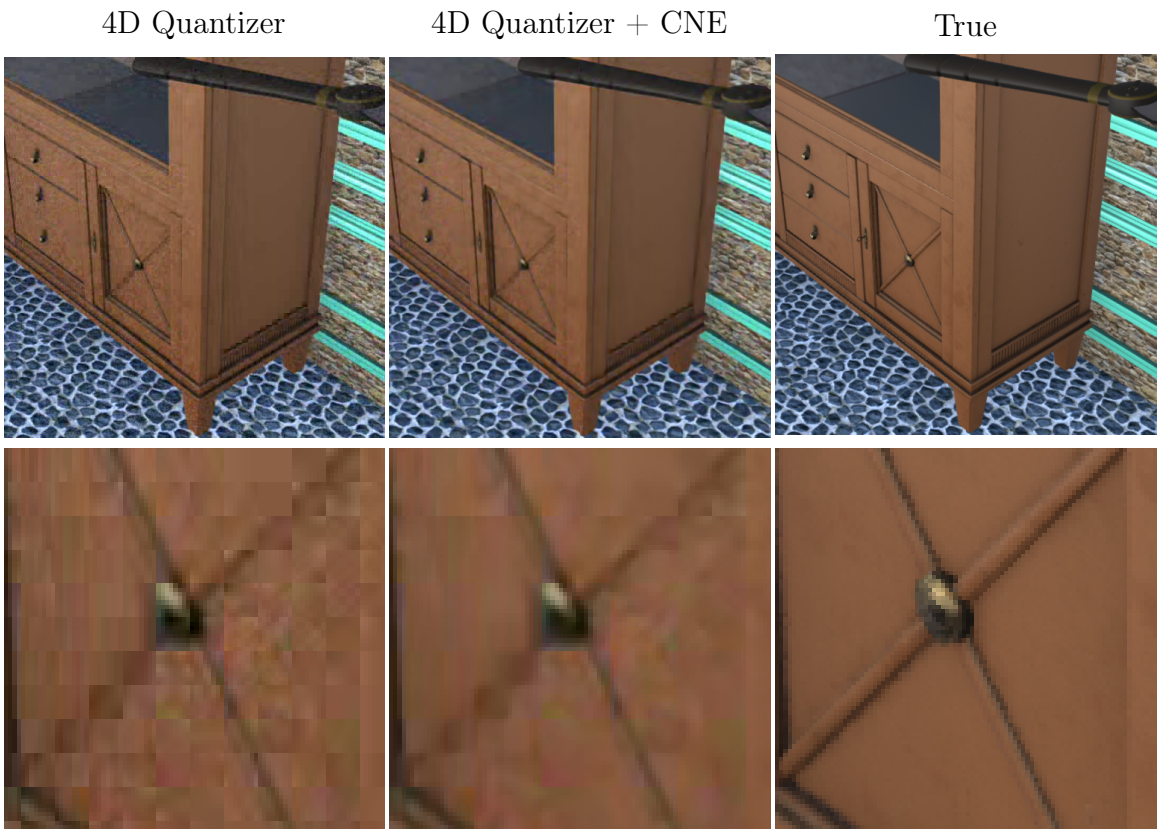


Figure D.7: Example of a scene compressed with our models at a typical bitrate.

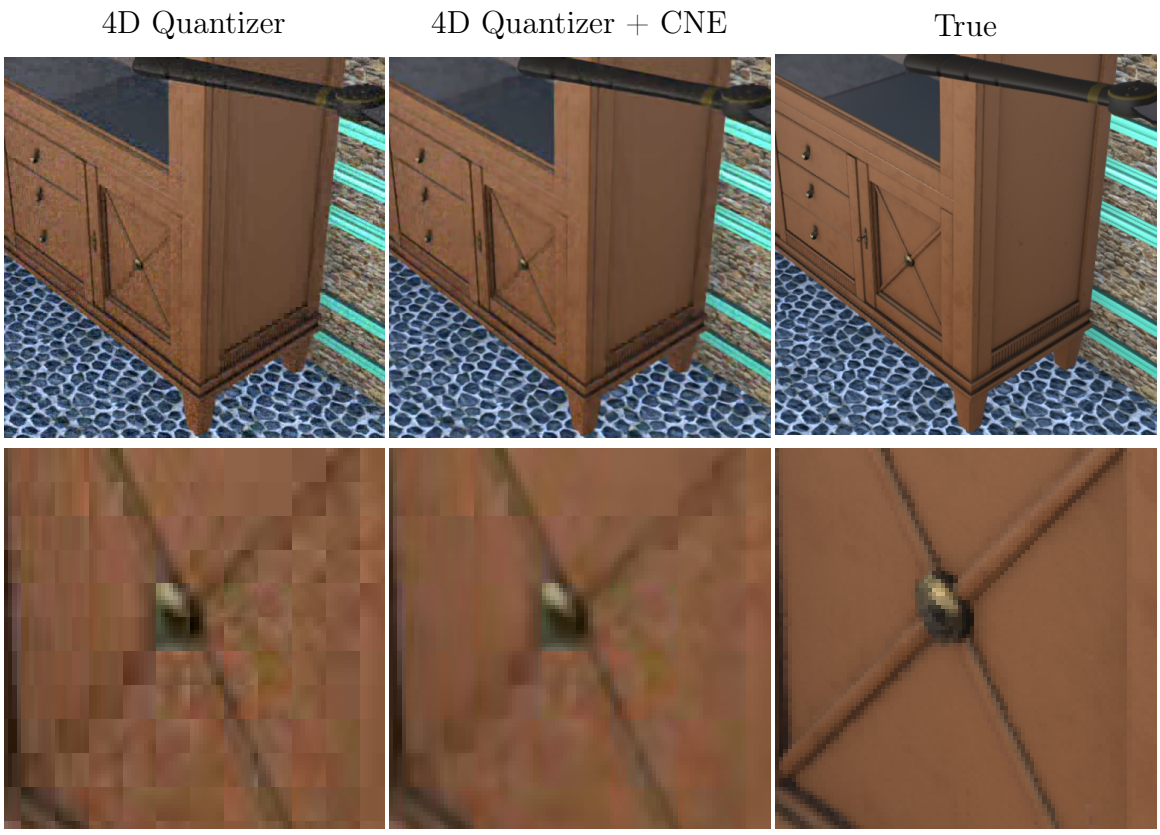


Figure D.8: Example of a scene compressed with our models at a low bitrate.

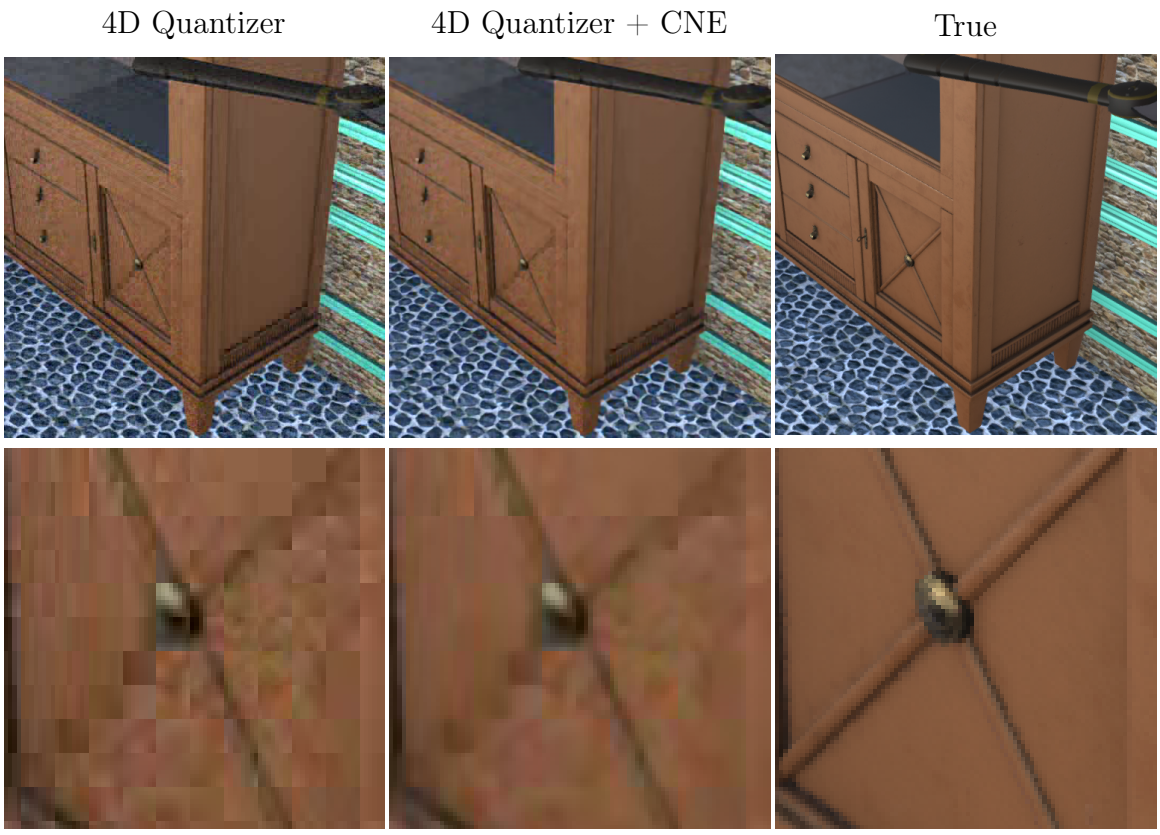


Figure D.9: Example of a scene compressed with our models at a very low bitrate.